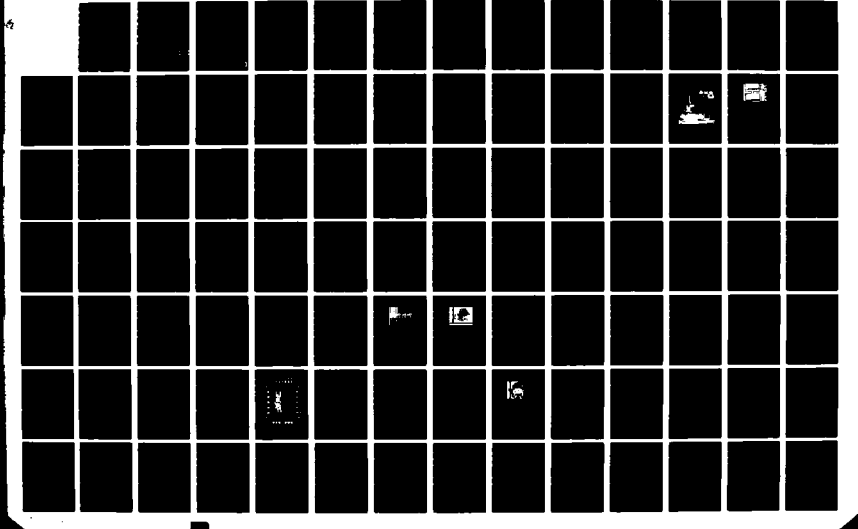
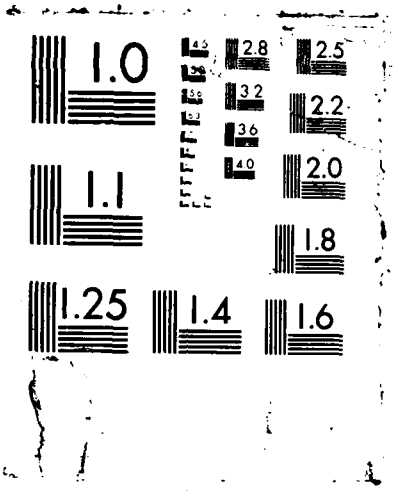
AD-A189 590 LASER PROGRAMMING INTEGRATED CIRCUITS(U) AIR FORCE INST 1/2
OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
C S SPANBURG DEC 87 AFIT/GE/ENG/87D-62

UNCLASSIFIED F/G 9/1 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

LASER PROGRAMMING INTEGRATED CIRCUITS

Thesis

Craig S. Spanburg
Captain, USAF
AFIT/GE/ENG/87D-62

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/87D-62

LASER PROGRAMMING

INTEGRATED CIRCUITS

Thesis

Craig S. Spanburg
Captain, USAF
AFIT/GE/ENG/87D-62

AFIT/GE/ENG/87D-62

LASER PROGRAMMING INTEGRATED CIRCUITS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Craig S. Spanburg, B.S.

Captain, USAF

December 1987

ACKNOWLEDGEMENTS

This research effort could not have been completed without the help and understanding of several people. I would first like to thank my loving wife Melissa for all her support and patience. I would also like to thank my children, Mary and Nicholas, for their cooperation.

I am grateful to the other members of the VLSI group: Captain Keith Jones, Captain Dave Gallagher, Second Lieutenant Scott Hauser, Captain Gray Salada, and Captain Erik Fretheim for their assistance and encouragement.

A special thanks to my thesis advisor Captain Richard Linderman for his guidance and encouragement. I would also like to acknowledge the the members of my thesis committee Major Glenn Prescott, and Captain Steve Rogers. Lastly, I would like to thank Lieutenant Colonel James Mills for his advise.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF TABLES

ABSTRACT

Programmable circuits are building blocks that improve design efficiency. Each mask and field programming technique has its own set of advantages and disadvantages. This thesis considers laser cutting as an alternative programming technique.

This effort involves designing, assembling, and demonstrating a laser programming system. The laser system includes lasers and optics, a microprobe station, and a motion controller. Software for remote operation of the motion controller and for generating the cutting location coordinates is also developed during the research.

Two applications of laser programming, a 32-bit comparator and a serial multiplier, are examined. The design of the two chips illustrates laser programmable circuit design and layout considerations. Programming the fabricated 32-bit comparator demonstrates the laser system. Testing the comparator verifies the performance of a laser programmable circuit.

LASER PROGRAMMING INTEGRATED CIRCUITS

CHAPTER 1

Introduction

1.1 Background

Integrated circuit design innovations such as interactive layout editors and simulation software, and conveniences such as cell libraries and off-the-shelf components improve productivity and quality. Layout and simulation software eliminate the burden of time consuming and error prone manual methods. Cell libraries and off-the-shelf components are building blocks that allow design efforts to be concentrated on the problem of interest and not the supporting circuitry. The function of the cells and off-the-shelf components may be fixed, as are logic gates, or programmable, like programmable logic arrays (PLAs). The programmable cells and components enable the function to be customized for specific applications.

Programming can occur at the mask level for cells or in the field for off-the-shelf components [Mur82]. The family of read-only memories (ROMs) is an example of mask- and field-programmable components. A mask-programmable ROM is prepared by changing only one of the masks used for fabrication. The preparation of just one mask saves design time and the expense of making a new mask set. A field-programmable ROM (PROM) is an off-the-shelf component that a user can write information to once. To

store information, a fuse is blown at the selected locations. In both instances, the programmable ROM saves design and fabrication time and expense.

Laser cutting is a field-programming technique that is similar to making disconnections by blowing fuses. In this method, the integrated circuit is specially designed with laser programmable features. These laser programmable features consume less area than the circuitry supporting the other field-programmable techniques. This technique is better suited for insertion on chip because less area is used. The other techniques are better suited for use at the board level. Two applications for which laser programming could be useful are a comparator and a serial multiplier. The comparator performs 32-bit comparisons between an address bus and programmed address values. The programmed address could be implemented by a bank of loadable flip-flops or by hardwiring the programmed address on the chip. However, a variation of the flip-flop or hardwired technique, specified for this application, is to laser program the addresses into the chip after fabrication.

The serial multiplier is one element of a large multiplier array. The multipliers in the array are connected to an adder tree resulting in a pipelined bit-serial multiplication architecture. The Winograd Fourier Transform Processor multiplication uses fixed coefficients which are hardwired into the serial multiplier elements. These fixed coefficients could be laser programmed after fabrication.

Laser programming can also be used to select backup circuitry. When a section of circuitry fails, a redundant circuit takes the place of the bad circuit. The change occurs by cutting out the bad circuit and patching in the replacement. Thus, laser programming supports fault tolerance along with circuit customization.

1.2 Problem Statement

The objective of this thesis is to design and demonstrate a laser programming system that employs cutting only. A secondary objective is to design two laser programmable circuits. One circuit is a 32-bit comparator and the other circuit is a serial multiplier. As stated previously, the 32-bit comparator compares a dynamic address field to a laser programmed address field. The serial multiplier finds the product of a dynamic multiplicand and a laser programmed multiplier.

The laser system design problem includes identifying an appropriate laser, arranging the laser and optics, determining the parameters for laser cutting, and developing a user interface. The laser programmable circuit design problem involves designing laser programmable structures and incorporating them in the circuit design. Once the circuits are fabricated, the problem becomes directing the laser processing to the programming locations.

1.3 Scope

This thesis is limited to the design and laser programming of two circuits. The chemistry and physics of, and the procedures for, laser cutting integrated circuits are described in the integrated circuit technology literature. The procedures described in the literature are modified to accommodate the available resources. Thus, this effort is limited to the implementation of techniques to accomplish laser programming and does not attempt to characterize the chemistry and physics of the laser cutting. The techniques are also automated to allow selective laser cutting over an entire integrated circuit. The automation depends on the two individual designs and the rules developed during the initial experiments on a test structures. The laser programming is accomplished by cut-

ting, no connecting or writing is necessary.

The 32-bit comparator was specified as a component of a research effort conducted by a 1986 AFIT graduate 2Lt Larry French [Fre86]. He sought a means to avoid the time delay and space penalty of using 8 to 16 multiple pin packages to perform a set of four comparisons. The need for the serial multiplier originated in the design of a Finite Impulse Response (FIR) filter [Lin86]. He specified an array of serial multipliers with programmable coefficients. Consequently, this research focuses on combining laser programming with these designs in contrast to justifying these designs.

1.4 Summary of Current Knowledge

At least one manufacturer makes laser systems for production laser programming. The Florod systems incorporate a cutting laser, a microprobe capability, an antivibration table, a video camera and monitor, and PC programmability. These systems are used for resistor trimming, substrate hole drilling, substrate cutting, and substrate marking. The basic version requires manual operation, but the advanced versions allow automation of the laser parameters and beam movement.

In the literature, laser processing of integrated circuits is covered in depth, but information on applications for laser programming is limited. The two applications that are publicized are a laser programmable gate array and wafer scale integration. OrBach developed a micromachining laser system for gate array prototyping [OrB87]. Many wafer scale integration projects using laser programming are underway. Raffel used laser links and cuts to incorporate redundancy on a wafer-scale digital integrator [Raf80]. Other than these two applications, the literature is limited to characterizations of laser processing and demonstrations of fault correction using laser processing.

1.5 Approach

The two goals are the implementation of laser cutting techniques and the design of the two laser programmable circuits. First, the laser system is specified and the limiting parameters, such as the size of the laser cut, are determined. With the knowledge of the limiting factors of the laser system, a test structure and the 32-bit comparator are designed and fabricated first for use as a test chip. After the experiments are conducted on the test chip, the serial multiplier is designed taking into account the results of the experiments on the test chip.

The experiments on the test chip are accomplished as follows. First, the procedure to align the circuit features, focus the laser, and move the chip is determined. Second, the types and power levels of lasers for cutting described in the literature are verified. Third, the most successful laser parameters are used to manually cut a test structure on the test chip. Fourth, the circuit on the test chip is programmed by laser cutting. This step includes automating the movement of the x-y stage along with cutting connections within the laser programmable comparator. The results of this step could lead to the recommendation of design changes in the comparator and serial multiplier.

1.6 Materials and Equipment

This thesis requires the use of the Very Large Scale Integrated circuit (VLSI) design tools. Magic is an interactive editor used to layout the circuits. Mextra converts the data generated while using Magic into a simulatable circuit description. Esim is used to perform switch level simulations of the circuit. Spice is used to model parameters such as speed and power dissipation.

The computers used to run the tools are two ELXSIs and two Sun workstations. The terminal used with the ELXSIs is an AED767 with digitizing pad as the graphics device. The laser and optics needed for this research are described in detail in Chapter 3. The laser system also requires a microprobe station, an antivibration table, stepping motors, and a motion controller. Lastly, a MicroVax computer is used as a user interface and file storage device for the motion controller.

1.7 Sequence of the Presentation

Chapter II is a detailed analysis of the problem of laser programming and of the equipment needed for laser processing. In Chapter III, the findings from the detailed analysis of the problem are used to design the laser system. In Chapter IV the design and VLSI implementation of the two laser programmable circuits is presented. An evaluation of the laser system and the two laser programmable chips is given in Chapter V. Chapter VI contains the conclusions and recommendations resulting from this effort.

CHAPTER 2

Detailed Analysis of the Problem

2.1 Overview

Circuit designers use cell libraries and off-the-shelf components to save design time and effort. These cells and components may work as is or may need to be customized to work in the specific application. The changes required to adapt the cell or component to the application may be referred to as programming or personalization. Structures can be programmed before or after fabrication.

Existing programming techniques have limitations. Mask-programming can only occur before fabrication. A new fabrication run is required for each application. Present field-programming requires special circuitry on chip to blow fuses or to electrically alter the logic value of the targeted location. The special circuitry takes up a significant amount of the area on a field-programmable chip [Mur82].

Laser programming is a technique which does not require a fabrication run for each application, or area consuming special circuitry. The component can be designed to be programmed by a series of cuts. This chapter describes the problem of making the disconnections with a laser and the problem of designing programmable features which employ disconnecting. In addition, this section considers two applications of laser programming.

2.2 Laser Programming Solution

Laser programming requires the consideration of several issues. The most obvious issues are laser cutting and the design of programmable features. However, the dimensions of the features that exist on an integrated circuit chip result in observation and laser aiming problems that are difficult to solve. The means to locate the sight of the disconnection and to steer and focus the laser beam to dimensions on the order of microns is necessary.

This section deals with laser cutting and programmable features, the laser system, and the automation of the laser programming procedure. In conjunction with the laser processing and programmable features, the application dependencies of the laser programmable features are explored. Included in the look at the laser system is a presentation of the suitable types of lasers, beam directing optics, and viewing aids required. Last, but very important to this effort, the probability of needing to make many cuts on a chip leads to the consideration of automating the laser system.

2.2.1 Laser Programming

A review of the integrated circuit literature identified the three cutting techniques covered in Section 2.2.1.1. In addition, these articles gave insight into how precise lasers can be aimed and what beam sizes can be achieved. However, there is little discussion on how the disconnections are being used. One of the articles did a good job of describing the laser cutting mechanism [Yam85].

Most of the laser energy is absorbed by the metal. The encapsulated metal melts and evaporates resulting in high vapor pressure inside the encapsulation. Eventually, the pressure builds to the point where the encapsulating layers are explosively removed from

the top of the metal. The metal vapor and molten metal explosively leave the area resulting in a disconnection.

2.2.1.1 Laser Processing. Hiroshi Yamaguchi developed a process to cut aluminum using a nitrogen pumped dye laser with a wavelength of 510 nm and a pulse duration of 12 ns [Yam85]. He chose a single-shot method over a spot-scanning method. The scanning was found to cause damage to the substrate in areas where the aluminum had already been removed by the proceeding overlapped pulses. The two single-shot aperture shapes considered were circular and rectangular. The rectangular single-shot technique was used because the rectangular beam could be limited to the width of the aluminum stripe.

Various widths of 1.75 μm thick aluminum were cut over a 2.45 μm insulating layer of silicon dioxide. Single laser pulses with intensities of 2.22 x 10^9 W/cm^2 were found to be most successful. The power densities ranged from 0.35 to 0.8 x 10^9 W/cm^2 for cuts through a 0.1 μm silicon dioxide layer without damaging the substrate. As the silicon dioxide thickness increased, the power density range also increased. For 2.0 μm silicon dioxide, the power density range was from 0.35 to 1.5 x 10^9 W/cm^2.

Matching the width of the rectangular beam to the stripe width did not work as expected. Defocusing occurred at the edges of the beam and aluminum remained at these edges. Consequently, the width of the aperture was varied to find the optimum rectangular beam width. In fact, a 4 μm wide stripe required beam widths between 4.2 and 5.3 μm. Too wide a beam damaged the substrate adjacent to the stripe and resulted in shorting to the substrate.

G. Koren demonstrated that XeCl excimer laser assisted aluminum etching in chlorine gas is a very efficient process for cutting larger areas than Yamaguchi [Kor85]. The large area cutting was accomplished by using a broad uniform beam and a patterning mask. The etching had two independent steps - chemical reaction and photoablation. First, the chlorine gas reacted with the aluminum to form an aluminum chloride layer. Next, the laser pulse ablated, or eroded, the aluminum chloride layer. If the chlorine gas pressure was too high, then the ablated material collided with the many chlorine gas molecules and redeposited on the aluminum. If the chlorine gas pressure was lower, then there was enough chlorine for the chemical reaction but not enough to hinder photoablation.

Using the laser, an irradiation cell, and quartz optics, etch rates as high as 1 μm of thickness per pulse were obtained. Pulses 40 ns long with energies of 0.4 J were used to uniformly illuminate a 2x2 mm^2 area. However, first the native oxide was removed from the surface by using four 3.4 J/cm^2, 40 ns laser pulses at 1 Hz with 0.4 Torr chlorine gas in the irradiation cell

Daniel Ehrlich described laser microchemical techniques for cutting at lower power densities [Ehr84]. He used a microscope, a 488 nm argon ion laser, and a computer controlled x-y stage. Position resolution of 0.25 μm was obtainable with the optical system. The laser beam was focused to a 3 μm spot using f/8 refractive optics. The pulse length of the laser was controlled by a 1 ms resolution shutter.

Cutting the aluminum was achieved by local heating of the substrate (200 ° C) with the laser beam. The substrate was emersed in a 0.15% KCr : 10% phosphoric-acid / 90% nitric acid etchant which was activated by the heat of the substrate. The solution was cooled to confine etching to the area of the beam Etch rates under illumination

were 106 times faster than in the area not under illumination. Resolution of 1.5 μm was obtained, and the low laser power levels resulted in the underlying layers not being damaged.

In summary, these three laser cutting techniques were representative of the literature as a whole. A laser was either focused to a spot and used to cut or a pattern was etched using a broad beam and a mask. Power densities and pulse lengths varied article to article. Also, chemical etchants were used in some techniques but not others.

2.2.1.2 Florod Laser Cutter The Florod system was demonstrated to us by a technical representative of the company. The basic system is shown in Figure 2-1. The system contained a 750 W Xenon laser that produced a 1 μs pulse. A rectangular aperture control permitted 3 to 10 μm wide cuts. The laser was targeted by a bright light that passed through the same aperture as the laser. A cut made by this system is shown in Figures 2-2. The operation of the Florod Laser Cutter was totally manual. The operator must find the cutting location, adjust the aperture, and pulse the laser. This system is similar to the one described by Yamaguchi. Both systems have pulsed lasers and adjustable rectangular apertures. Neither system required chemical etchants to assist the cutting.

2.2.1.3 Laser Programmable Features. The purpose of the programmable feature is to represent a logic high or low to a node in the circuit. The most direct method obtaining this condition is to run a line from power to ground and cut out the undesired connection as shown in Figure 2-3a. However, this technique requires that a cut be made at all programmable locations. For the average application this would require twice as many cuts as necessary thereby increasing the programming time by a

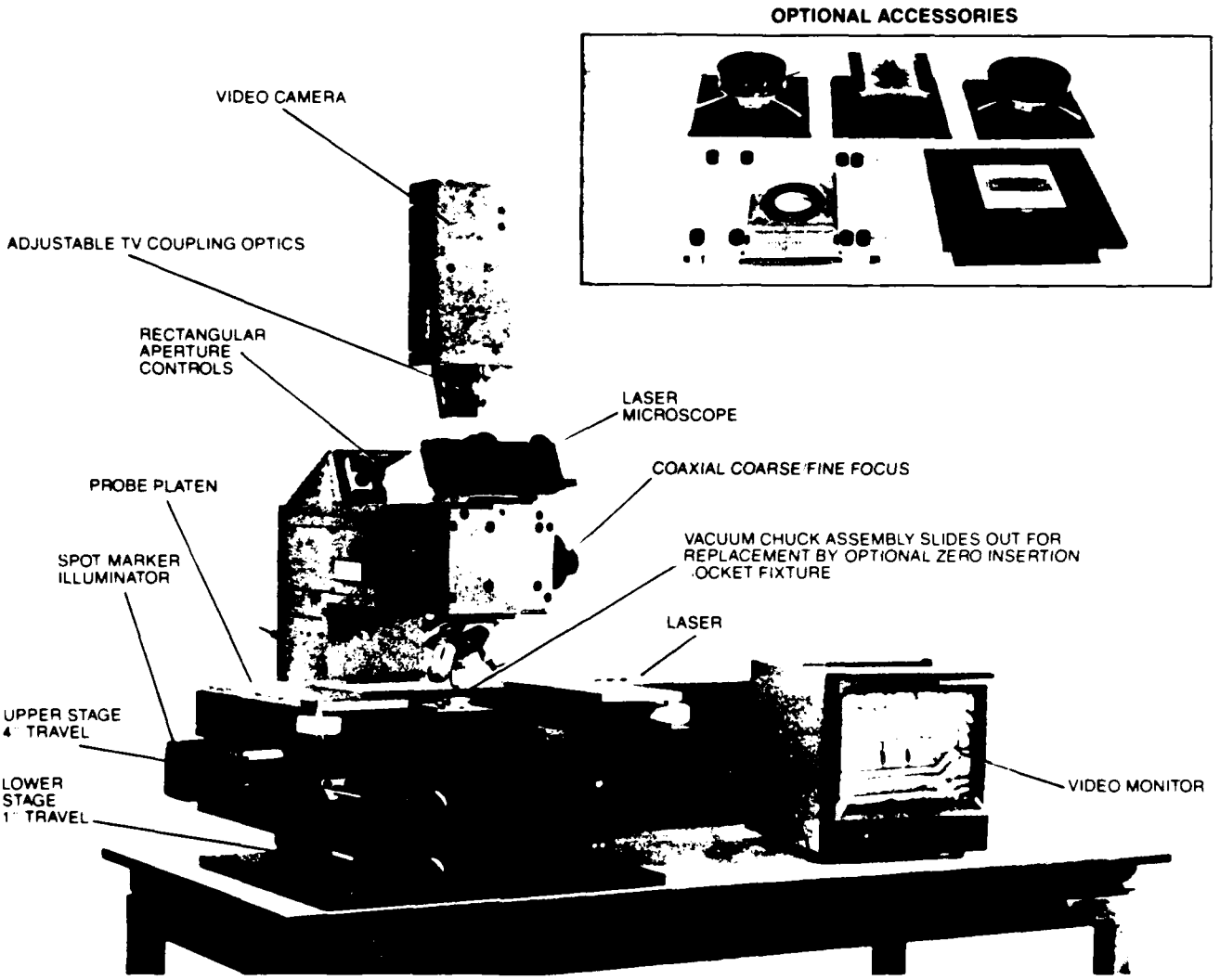


Figure 2-1 Florod Laser Cutter [Flo86]

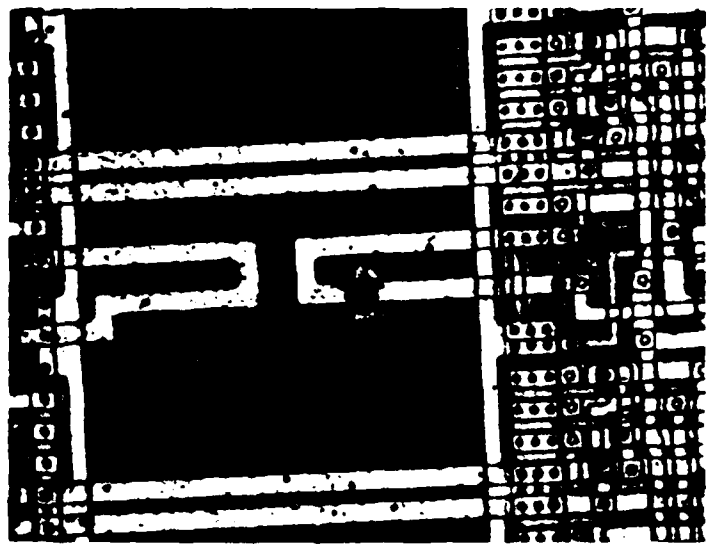


Figure 2-2 Florod Laser Cut

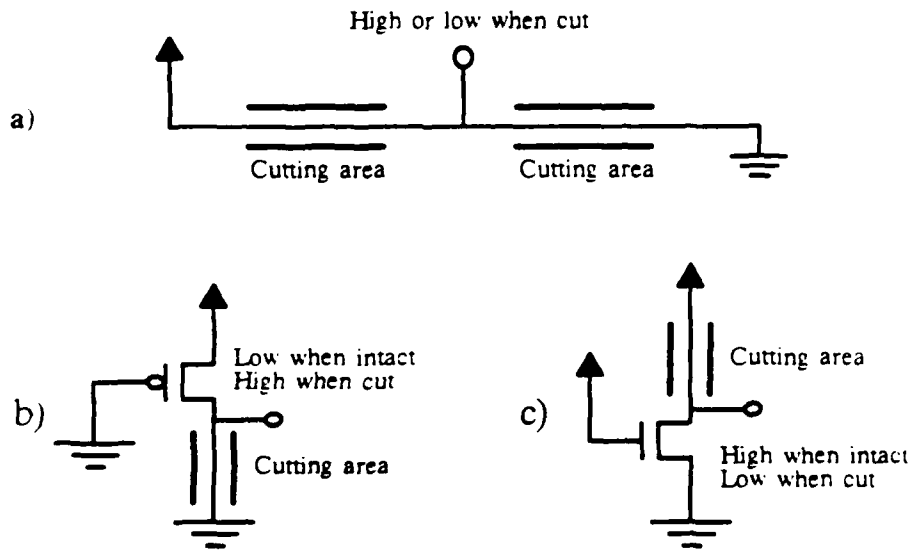


Figure 2-3 Laser Programmable Features

factor of two. More serious, if any cut is not successful a short from power to ground would result.

The advantage of the direct connection is the lower static power dissipation. The alternative is to short a high impedance MOSFET between power and ground as shown in Figures 2-3b and 2-3c. Figure 2-3b is an npn MOSFET with its gate tied to ground and the source and drain connected to power and ground. When intact, the programmable node is at logic low and power is dissipated, when a cut is made in the designated area the node goes to logic high and no power is dissipated.

Figure 2-1c is a pnp MOSFET with its gate tied to power and the source and drain connected to power and ground. When intact the programmable node is at logic high and power is dissipated. When a cut is made in the designated area, the node goes to logic low and no power is dissipated. For both MOSFET types, the gate can be made long and narrow so that very little current flows and very little power dissipation occurs when the connection between power and ground is left intact. However, if a very large number of these programming transistors are used, then the power dissipation may become a problem.

2.2.1.4 Applications. These programmable features can be used in existing applications such as ROMs, programmable logic arrays (PLAs), and gate arrays. The programmable transistors could be used as nodes in a ROM or PLA. The logic value of the nodes could be selected by laser programming. However, special applications must be considered in order to further justify this technique. One such application is a comparator. Figure 2-4a shows a programmable node being XORed with a dynamic node. The output would be logic low when the inputs match and logic high when the inputs do not

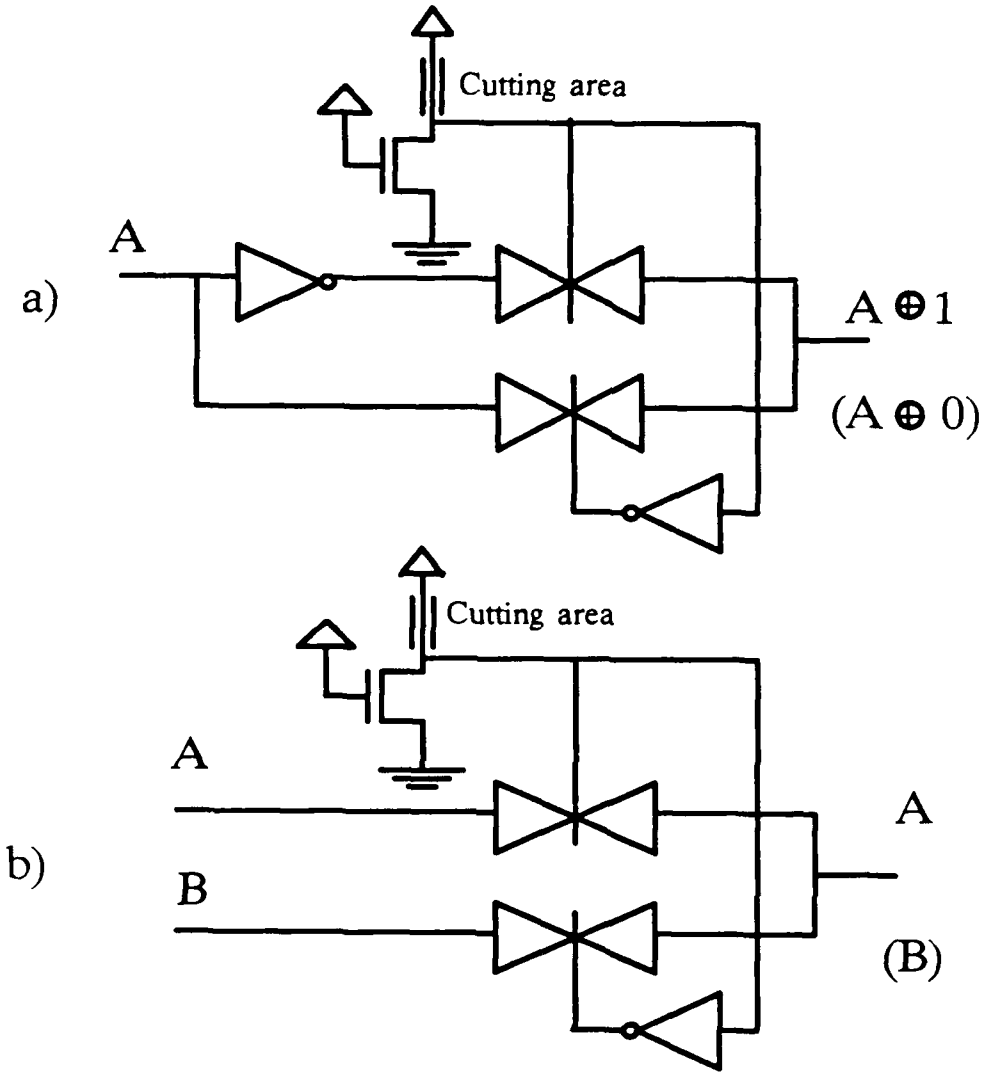


Figure 2-4 Laser Programmable Feature Applications

match. Another application is a multiplexer. Figure 2-4b shows a 2:1 multiplexer with the output being selected by the programmable transistors. This could be used to select between a circuit and its spare to provide redundancy to guard against defects and failures.

We see laser programming as an integral element in the rapid prototyping of application specific processors (ASPs) [Gal87]. The processor design implements only test microcode in the ROM so the remainder of the ROM is available for the application specific microcode. A large number of ASPs could be fabricated, tested, and put on the shelf for use as applications arise. Once an application is specified and the microcode is determined, a microcode assembler developed here at AFIT [Hau87] can generate the information needed to program the ROM. The laser system translates the assembler output and performs the required programming. The design, fabrication, and testing time to prototype ASP reduces from months to weeks or days.

2.2.2 Laser System

The laser system needs to solve the problems of steering the laser beam, focusing the laser beam to the desired location, and viewing the laser beam and the cutting area. The articles discussed in Section 2.2.1.1 refer to the laser types and power levels necessary to perform laser programming. Another consideration is the requirement to steer and focus the laser beam. Also, a method to view the aiming and the laser cutting at high magnification is needed.

2.2.2.1 Laser Types and Optics. The lasers mentioned in the literature have wavelengths of from 308 to 510 nm. They included an excimer, nitrogen pumped dye, and ion lasers with pulse durations from 12 ns to 1 ms. The power density required to cut aluminum on silicon was described to be on the order of 10^9 W/cm^2. The beams were able to be focused to spot sizes as small as 1.5 μm. Consequently, a laser with similar parameters and optics capable of steering and focusing the laser are necessary to cut with a laser.

The wavelength of visible light ranges from 400 to 700 nm so the laser light used to cut is not necessarily visible. As a result, some method is needed to show where the invisible laser is aimed. The options are an intensified non-laser light or a visible laser. The intensified non-laser light shows up brighter than the background light and the visible laser provides a colored spot to identify where the invisible laser is aimed. The laser light is steered using various optical components.

Lasers produce diverging beams which may need to be conditioned by a beam expander. The beam expander increases the diameter and collimates the beam as shown in Figure 2-5a. A collimated beam's light waves are parallel instead of divergent or con-

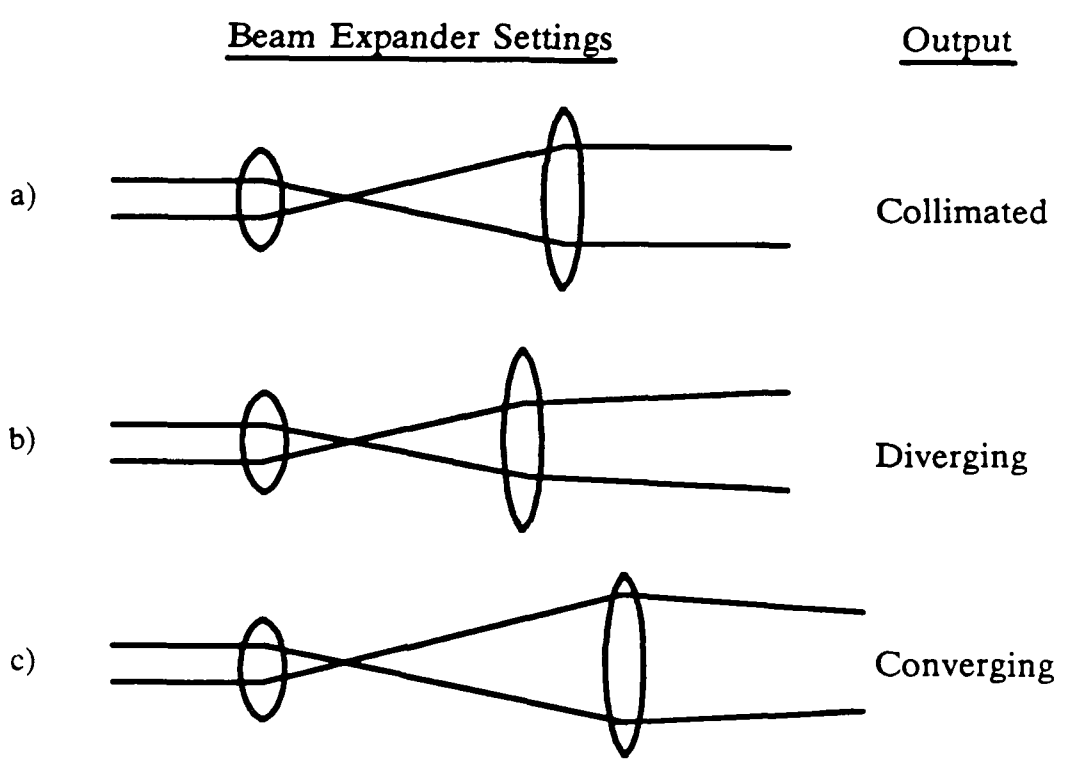


Figure 2-5 Beam Expander Outputs

vergent. By slightly increasing the distance between the lenses in the beam expander, the output beam becomes slightly convergent. Figure 2-5c illustrates how a converging beam is produced by a beam expander. At some distance away from the beam expander, the output beam diameter becomes smaller than the original beam diameter. As a result, the laser's power is retained and a degree of magnification is achieved from the beam expander. Since the beam is only slightly converging, the beam does not come to a focus before it reaches the other necessary optics.

Laser beam steering optics are generally wavelength and power sensitive because of reflective coatings. Mirrors reflect a high percentage of energy of a band of wavelengths but a lower percentage of energy outside that band. To merge the invisible cutting laser with the visible light, a beam splitter must be used. Beam splitters are also wavelength and power sensitive. The beam splitter reflects a high percentage of a range of wavelengths while allowing other wavelengths to pass. On the other hand, lenses used to focus lasers are not generally as wavelength or power sensitive because they do not require reflective coatings.

Lenses pass nearly all of the incident energy. However, at high power densities the small percentage of energy absorbed may be enough to damage the lens. Therefore, the composition of the lens affects the power sensitivity as does the coating. Some lenses are composed of pieces that are glued together. The glue may burn at certain combinations of laser wavelength and power density and damage the lens.

In summary, the cutting laser and aiming light source parameters must first be fixed. Next, the beam expanders, mirrors, beam splitters, and lenses may be selected knowing the laser parameters. Once, the cutting and aiming source and optics are acquired and set up, a method is needed to be able to view at the magnification required

to observe feature sizes on the order of 1 μm.

2.2.2.2 Microscope. A microscope must be used to view laser cutting because of the dimensions of the features. When the microscope is in place to view the cutting, it is most likely in the way of the cutting beam. Therefore, the microscope may have to be moved away during cutting or the cutting beam may have to be passed through the viewing optics. As a result, the optics used to magnify the image may be used to focus the cutting and aiming beams.

To facilitate the laser beams, the optics inside the microscope must meet the specifications required to pass the cutting and aiming light beams. The microscope must have a port for the cutting and aiming light beams, viewing, and the illuminating light source. The microscope must have two separate position adjustment mechanisms: one to position the microscope relative to the cutting and aiming light sources, and one to position the specimen relative to the microscope.

2.2.2.3 Video Camera and Monitor. In order to make viewing more convenient, a video camera and monitor could be used. Viewing through eyepieces for long periods of time is uncomfortable and tiresome. The video camera could be attached to the viewing port and the cutting could be viewed on a monitor. The operation could also be videotaped for quality control purposes. Moreover, an operator's performance could be evaluated or the technique could be demonstrated at a later time or at a more convenient location with the video tape. If a series of cuts was automated, then the quality of a programming run could be evaluated using the videotape.

2.2.3 Automation

Once the problems associated with the laser system are overcome, the probability of needing to make multiple cuts to program a structure becomes the next obstacle. If it took 10 seconds to manually find and cut a programmable location, then it would take longer than 16 minutes to manually find and cut 100 programmable locations. In order to make laser programming more than an academic exercise, the procedure must be automated. To facilitate automation the programmable structure should be a regular structure to allow step and repeat operations or must be at least a well defined structure so that the programming locations may be easily found.

The considerations associated with automation are the hardware, the required accuracy and repeatability, and alignment. The hardware must provide computer control of the movement of the specimen and be compatible with the mechanisms used to drive the stage upon which the specimen sits. Position accuracy and the cumulative error affects must be established. Finally, the axes of movement must be aligned with the axes of the specimen layout.

2.2.3.1 Hardware. The hardware necessary to automate the movement of the specimen are a controller, motors, belts, and pulleys. The controller governs the movement of the motors by manually entering commands or by executing programs in memory. The belts and pulleys connect the motc s to the mechanism used to move the specimen stage. The controller memory must accommodate the number of possible programming locations or accept input from remote devices. The series of steps required to laser program a structure is referred to as a program.

In addition to manual entry of a program, the controller should have sufficient memory to store a library of programs or the capability to read and write programs externally. The external read and write could be accomplished by a disk drive or a standard interface to a computer with sufficient memory. The motors need to be capable of moving fractions of a degree of a revolution with minimal torque. The belts must not slip on the pulleys and the pulleys must be reasonably round and concentric to consistently transfer the movement of the motor to the specimen stage control mechanisms.

2.2.3.2 Accuracy and Repeatability. The movement of the stage must be accurate to within one-half of the overlap of the cutting laser beam over the feature to be cut as illustrated in Figure 2-6. However, the error associated with a series of movements could easily add up and result in inaccurate positioning, thereby limiting repeatability. In addition to the accuracy of the movement during the execution of a program, the initial location and relationship between the axes of specimen movement and layout need to be considered.

2.2.3.3 Alignment Marks. To assure the programmed movement begins with no error, the starting location must be found on the specimen and the programmed movement must occur on the same axes as the layout. Presently, the axes alignment must be done manually by iteratively scanning across the specimen horizontally and vertically to the ends of the layout and then making the necessary rotation corrections until the axes align. Then, the starting position is manually found. This is not a considerable limitation for single programmable structure applications.

However, parallel research into reverse engineering integrated circuits being conducted by Capt Erick Fretheim at AFIT may result in automated alignment using align-

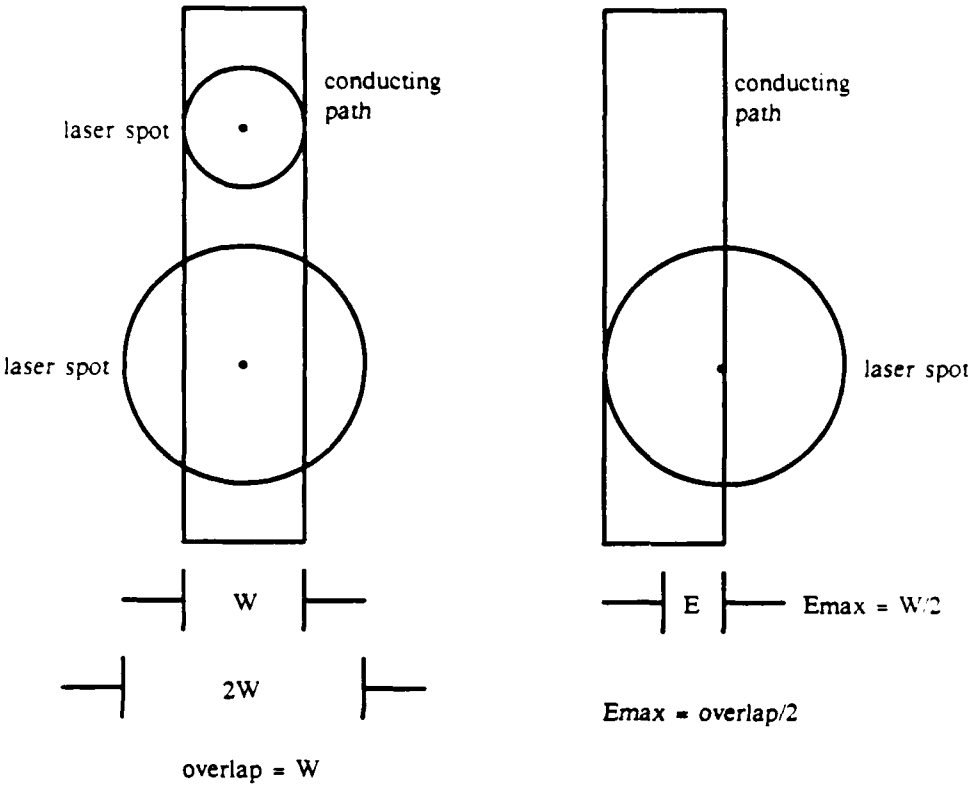


Figure 2-6 Pointing Accuracy and Repeatability

ment marks. Pattern recognition techniques are used to find the alignment marks. The coordinates of the alignment marks are then used to translate the coordinate system of the program to be executed. This feature would be particularly useful for applications in which several independent programs need to be executed sequentially on the same specimen. Likewise, positioning error could be corrected by realignment on marks at the cutting sites. Position error correction may be required after every movement or after a number of movements.

2.3 Design Rules

Once the cutting laser, the viewing system, and the automation system are specified, the combined effects must be considered to determine design rules. The cutting laser and viewing system affects spot size. The spot size drives the accuracy of the automation system. Since spot size and automation accuracy are implementation dependent, they are not specified for the consideration of design rules.

2.3.1 Laser System Considerations

The spot size of the cutting laser should be at least as large as the width of the lines or multiple cuts must be made across the entire line. Automation errors require cutting coverage to be larger than the line width as illustrated in Figure 2-6. For example, to cut a 3 μm wide line when the possible error is 1 μm in all directions, the cut needs to be 4μm wide. In addition, no other circuit features may be within the possible cutting area. For example, if the cut is 20 μm wide and the possible error is 5 μm in all directions, then no other feature should be within 15 μm of the cutting area.

2.3.2 Circuit Design Considerations

The layout of easily accessible and regular programming locations is the primary concern when designing laser programmable circuits. Laser cutting should have no effects on the operation of a circuit other than to cause the desired disconnections. When accomplished properly, laser cutting should not affect the operation of adjacent structures. Debris from the microexplosion caused during cutting may fall on nearby structures but the passivation layer provides electrical isolation from the debris.

2.4 Laser Programmable Circuits

As stated previously, many applications exist for which laser programming is useful. Two applications have been chosen to demonstrate laser programming. One is a comparator which performs a 32-bit comparison between an address bus and a static address value. The problem faced is how to implement the static address. Instead of fabricating a new chip for every implementation, the static address could be realized on a generic programmable design through laser programming. The static address could be made programmable using one of the programmable features shown in Figure 2-3. Then, the laser system could be used to laser program the static address onto the chip.

The other application is a serial multiplier cell in which the multiplier is static. The multipliers are normally hardwired or read in with the multiplicand into the multiplier. The problem faced by this application is how to implement the multiplier. Instead of building a cell for every possible coefficient, the multiplicand could be realized through laser programming a generic structure. The designs of the laser programmable 32-bit comparator and serial multiplier are two of the topics in Chapter 4.

CHAPTER 3

Laser System Design

3.1 Introduction

The analysis in Chapter 2 considered the several components required in a laser programming system. One component is a laser that cuts aluminum conducting paths on a silicon substrate. Another component is a microscope to focus the laser and to view the cutting. A motion controller permits automation of the cutting. The analysis also investigated current laser cutting parameters.

The laser power density required for cutting is approximately 10^9 W/cm^2 for 1 ns to 1 μs pulses or 10^5 W/cm^2 for 1 ms pulses. An aperture shapes the beam so the spot size can be varied without changing the power density. Circuit dimensions on the order of microns necessitate beam focusing and viewing with a microscope. An automation system improves programming efficiency for circuits with multiple programming locations.

This chapter presents the design of the laser system and a description of the components. Lasers and optics from the AFIT Electro-Optics lab, microprobe equipment from the AFIT Microelectronics Lab, and video equipment from the AFIT Signal Processing Lab combine to form the laser system.

3.2 Lasers and Laser Optics

The laser system requires a cutting laser and another light source to aid in aiming the cutting laser. A low power, visual laser is the light source chosen to aid in aiming

the cutting laser. In this section, the lasers, laser optics, and arrangement of the lasers and optics are described.

3.2.1 Visual Laser

The visual laser selected for this system is a Helium Neon laser which operates at a wavelength of 632.8 nm and an output power of 0.5 mW. The beam diameter is 0.59 mm which can be magnified to 10 μm. This is a red continuous wave laser that has a shutter to block the light when the laser is not needed.

3.2.2 Cutting Laser

The cutting laser selected for this system is a neodymium-doped yttrium aluminum garnet (Nd:YAG) laser which operates at a wavelength of 1060 nm and output energy of 0.1 J. The beam diameter is 6.35 mm which can be magnified to 10 μm. This is a pulsed laser which can put out single 20 ns pulses or 1, 5, or 10 pulse per second repetitions. The combination of the 0.1 J output energy, 20 ns pulse length, and 6.35 mm beam diameter result in a power density of 4×10^6 W/cm^2. Thus, the beam must be magnified between 90 and 375 times to achieve the required power density range (0.35 to 1.5×10^9 W/cm^2 from Section 2.2.1.1), but this is less than the 635x needed to reduce the beam to a 10 μm spot. An aperture should be used to allow the proper combination of power density and beam size. However, there is no aperture to control the beam size to the dimensions required for this research or power controls on the laser to limit the power density.

3.2.3 Laser Optics and Arrangements

There are two arrangements for the laser and laser optics [Mil87]. Figure 3-1a shows the ideal arrangement which uses beam expanders to produce converging beams

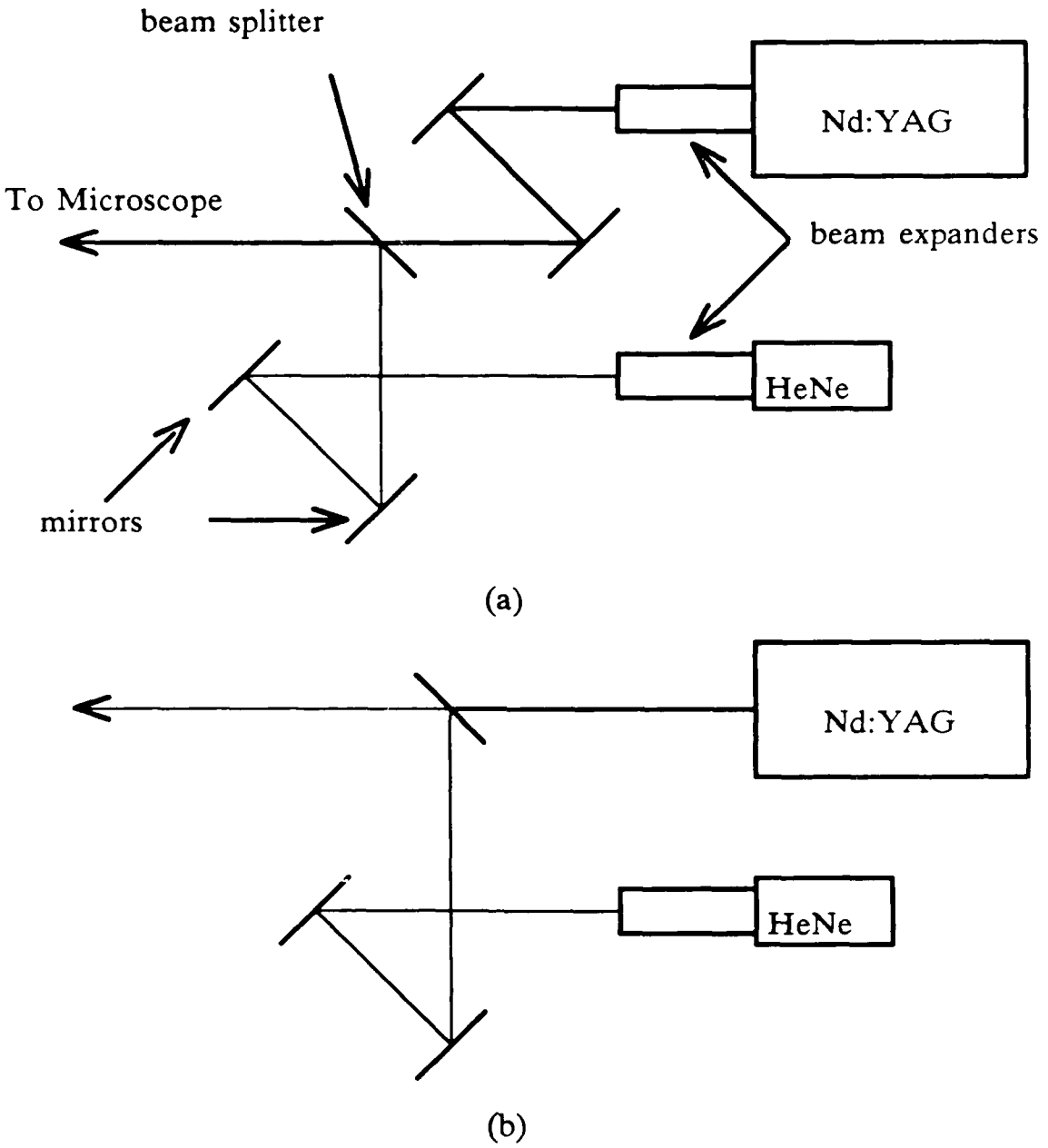


Figure 3-1 Lasers and Laser Optics Arrangements

from both lasers. These beams converge to diameters smaller than the original diameters. As a result, power loss from divergence is eliminated, power density is increased,

and some magnification of the beams is achieved. After the beam expanders, mirrors steer the beam toward the microscope optics. The mirrors also provide fine adjustment of the angle and location of the beams. After the mirrors and before the microscope optics, a beam splitter is used to merge the beams together. Unfortunately, no mirrors or beam expanders were available at AFIT for the Nd:YAG laser so an alternate arrangement was used.

Arrangement two, shown in Figure 3-1b, has no beam expander or beam steering mirrors for the the Nd:YAG laser. This arrangement allows initial testing of the laser system. Mirrors are only needed for the HeNe laser, but a beam splitter is still needed to merge the beams. All the optics for the HeNe are rated for use at 632.8 nm and 0.1 mW of power. The beam splitter passes the 1060 nm Nd:YAG beam and reflects the HeNe beam.

3.3 Microprobe Station

The microprobe station consists of a microscope, an x-y translation stage, and a set of probes. The microscope is movable in the x, y, and z directions independent of the x-y translation stage. The x-y translation stage holds the specimen for viewing and probing. This section describes the features of the microprobe station.

3.3.1 Microscope

The microscope is a Bausch and Lomb MicroZoom with a quadruple nosepiece that accepts 2.25X, 8X, 25X, and 40X working objectives and a 2:1 zoom in the body. The x-y-z translation is 1 inch by 1 inch by 1 inch and the focus has coarse and fine adjustments. A fast lift handle allows lifting the microscope for access to the stage without adjusting the focus controls. The lasers enter the microscope body through the light

source port.

The reflective optical surfaces are not designed for 1060 nm light, and the manufacturer, Bausch & Lomb, has found that the existing coatings reflect 15% of 1060 nm light. The power density limits are not known by the manufacturer. Therefore, the initial power density must be low. With each incremental increase in the power density, the mirror at the input port where the cutting laser enters the microscope must be inspected to minimize any damage which may occur. This mirror normally reflects the light from the 30W light source and is not in the path of the image. Thus, spot damage to this mirror would affect the illumination of the specimen but not the transfer of the image. Heat build up is not a factor because of the short pulse length.

3.3.2 X-Y Stage

The x-y translation of the stage is 6 inches by 6 inches and the rotation control has a range of 360 degrees. The resolution of movement is 1.49 cm per revolution for the coarse adjustment and 25.4 μm per degree of revolution for the fine adjustment. The fine adjustment knobs are mounted on a 1/4 inch shaft and are easily detached and replaced by pulleys. The pulleys are connected with belts to stepping motors to remotely and automatically control the movement of the stage.

3.4 Controller

The controller executes programmed movement of the x-y stage in relation to the microscope. The controller connects to two drivers that power the two motors attached to the fine adjustment controls of the x-y stage. The controller may be programmed manually or programs may be loaded externally via an RS232 interface. This section identifies the hardware and software features of the controller, and the accuracy and

repeatability of the programmed movement.

3.4.1 Controller Hardware

A MITAS two-axis controller provides the programmed movement of the x-y stage with two stepping motors. The stepping motors are driven by separate power supplies that translate the commands from the controller into the current waveforms necessary to move the motor shaft. The shafts connect to the x-y stage fine adjustment mechanism with belts and pulleys. The controller has the following commands:

1. dump the memory
2. program controller
3. execute command
4. prints messages on controller display
5. dumps the current location
6. sets home of reference
7. execute grouping of program lines.

The controller is connected to a MicroVax via an RS232 interface to provide file support and remote operation.

3.4.2 Controller Software

A C program is used on the remote machine to access the controller. The C program is a user friendly interface to the controller and supports all remote commands. In addition, the program assists the user with reading and writing files. The controller only handles one line at a time so dealing with long input or output strings manually would be monotonous. The program inserts the controller commands and automatically reads or writes the user defined number of lines.

The C program treats the port the controller is connected to as a file. Thus, the channel is opened and closed like a file and all reads and writes are made through the file pointer. The data received from the controller may be easily written to a user designated

file. Conversely, the data sent from the controller may be read out of a user designated file. Consequently, a library of programs may be stored on the MicroVax. The main portion of the C program follows.

```
#include <stdio.h>

main()              /*   'main' opens port txa5:, display's the menu, gets */
                   /* a choice from the user, checks the validity of the  */
{                  /* choice, and calls the appropriate function.        */
                   /*   'main' loops until the quit command is issued.    */
int valid;          /* Upon the quit command, the MITAS session is ended,  */
char a, c;          /* port txa5: is closed, and the program is exited.    */
char choice = "A";
FILE *fp, *fopen();

fp = fopen( "txa5:", "r+" );
printf( "You have entered the mitas program." );
printf( "Press return then place the controller into the RS232 mode" );
printf( "at 600 baud." );
a = getchar();
fputc( a, fp );
fputc( '012', fp );
while( ( c = getc(fp) ) != 'n' ) printf("%c",c);
fputc( '015', fp );
while( ( c = getc(fp) ) != 'n' ) printf("%c",c);
printf( "Press <return> to continue." );
getchar();

while( choice != 'Q' ) {
    printf( "Select function" );
    printf( "D - dump mitas memory" );
    printf( "W - (where) current x and y coordinates" );
    printf( "P - program a line" );
    printf( "X - execute a move" );
    printf( "S - set current position as home" );
    printf( "L - execute program lines" );
    printf( "M - messages" );
    printf( "Q - quit" );

    printf(">");
    choice = getchar();
    getchar();
    valid = 0;

    if ( choice == 'D' ) {
            dump( fp );
```

```
            valid = 1;
        }

        if ( choice == 'W' ) {
            where( fp );
            valid = 1;
        }

        if ( choice == 'P' ) {
            program( fp );
            valid = 1;
        }

        if ( choice == 'X' ) {
            xecute( fp );
            valid = 1;
        }

        if ( choice == 'S' ) {
            set_home( fp );
            valid = 1;
        }

        if ( choice == 'L' ) {
            line_execute( fp );
            valid = 1;
        }

        if ( choice == 'M' ) {
            message( fp );
            valid = 1;
        }

        if ( choice == 'Q' ) {
            quit( fp );
            valid = 1;
        }

        if ( valid == 0 ) {
            printf( "Your choice is not valid, try again." );
            printf( "Press <return> to continue." );
            getchar();
        }

    }
}
```

Each of the seven remote controller operations are supported by the program. The dump and program are file operations whereas the other commands require user input data. The remainder of the code is included in Appendix A. Two other programs automatically generate programming locations. One program generates the cuts to program an address and the other program generates the cuts to program an encoded multiplier term.

An example of the arrangement of programmable locations is shown in Figure 3-2. For this example, two 8-bit addresses need to be programmed. The horizontal spacing is 100 units between cutting locations and the vertical spacing is 100 units between cutting locations. The addresses implement horizontally from the least significant bit on the left to the most significant bit on the right.

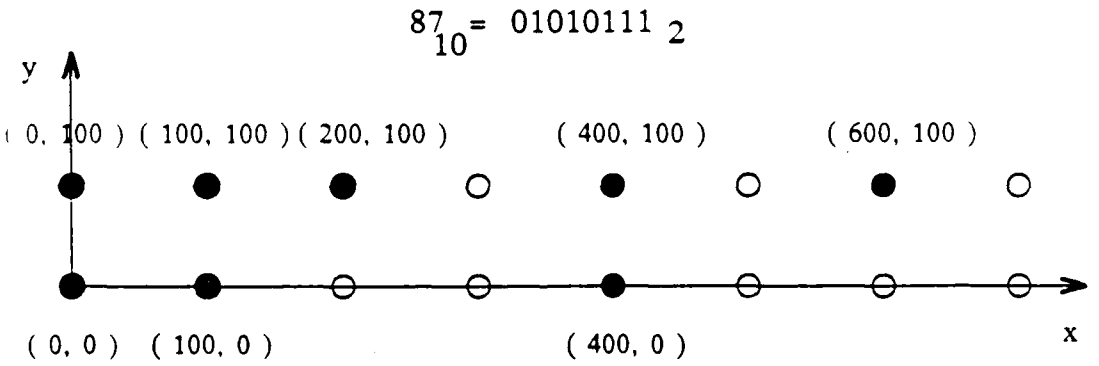


Figure 3-2 Programmable Location Software Example

The program generates the coordinate pairs for cutting locations. A cut is required where the address bit is a one. The number of addresses, address length, and address value are input by the user. The spacing may be changed by editing the program. Given two decimal address values 19 and 87, the binary equivalents are 00010011 and 01010111. Thus, three cuts are made in one row and five in the other. Next is a listing of the terminal session and the program.

```
You have entered the address location program.
Enter the name of the output file: 2x8


How many rows (addresses) are in your design? 2

How many columns (address length) are in your design? 8

Enter the decimal address for row 1: 19

The programming locations for row 1 are:
0 0
100 0
400 0

Enter the decimal address for row 2: 87

The programming locations for row 2 are:
0 100
100 100
200 100
400 100
600 100
[32]afitbsd cat 2x8
0 0
100 0
400 0
0 100
100 100
200 100
400 100
600 100
[33]afitbsd
```

```
/*****************************************************************************
* Program name: comparator
* Function: Generates the cutting locations for an address implementation.
* Date: 14 Sep 87
* Version: 1.0
*
*  This program converts a decimal address into the cutting locations used
* to implement the address. The number of address, address length, and
* address values are the input. The x-y coordinates of each cut are output
* to the screen and to a file.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

#include <stdio.h>

main()
{
    FILE *fpout, *fopen();
    int r, c, a, i, y, b, j, loc[32], k, x, error;
    char outfile[8];

    printf( "You have entered the address location program." );
    printf( "Enter the name of the output file:  " );
    scanf( "%s", outfile );
    fpout = fopen( outfile, "w" );

    printf( "How many rows (addresses) are in your design?  " );
    scanf( "%d", &r );
    printf( "How many columns (address length) are in your design?  " );
    scanf( "%d", &c );
    y = 0;

for ( k=1; k<=r; k++ ) {
    printf( "Enter the decimal address for row %d:  ", k );
    scanf( "%d", &a );
    x = 0;
    j = 0;
    error = 0;
    for ( i=1; i<=c; i++ ) {
        b = a % 2;
        a = a / 2;
        if ( b == 1 ) {
                loc[j] = x;
                   j = j + 1;
        }
        if ( i==c-1 && a>1 ) {
                   printf( "*** Address too large for number of " );
```

```
                printf( "columns given.  Try again. ***" );
                k = k - 1;
                error = 1;
            }
            x = x + 100;
        }
        if ( error != 1 ) {
                printf ( "The programming locations for row %d are:", k );
                for ( i=0; i<j; i++ ) {
                    printf( "%d %d", loc[i], y );
                    fprintf( fpout, "%d %d", loc[i], y );
                }
        }
        y = y + 100;
    }
    fclose( fpout );
    }
```

Similarly, the multiplier program generates the locations for the programmable coefficient. The output from both programs can be used in a Mitas program to control the movement of the x-y stage. Programs supporting any application can be written and used in conjunction with the Mitas controller to assist movement to the cutting locations.

3.4.3 Accuracy and Repeatability

The controller moves the motors in increments of 1/64 of 1 step. The step size of the stepping motor is 1.8 degrees and from section 3.3.2 the fine adjustment moves the x-y stage 25.4 μm per degree. The result is that the controller can move the x-y stage in increments of 0.714 μm. The motors are accurate to $\pm$ 3% of a step. This 0.0214 μm error may occur on every move and for every 360 degrees of revolution of the motor. Thus, the error is cumulative. Consequently, the accumulation of error must be considered when determining the laser spot size and design rules. Another source of error are the belts and pulleys from the motors and within the micromanipulator. The magni-

tude of the error attributable to the belts and pulleys is determined in the results chapter.

3.5 Other Components

The other components of the laser system are a movable table for the micromanipulator, a video camera, and a video monitor. The lasers, laser optics, and microprobe station would ideally be kept on an antivibration table. However, due to freedom of movement desired for the micromanipulator an antivibration table is not used for the microprobe station. A movable table aids the alignment of the microscope input port with the laser beams. The video camera and monitor provide convenient viewing of the specimen through the microscope.

CHAPTER 4

Laser Programmable Circuit Design

4.1 Design Rules

The design rules for laser cutting are a function of the laser system and the circuit design. The laser spot size and the laser system position accuracy directly affect the design rules. In addition, the layout of the circuit is dependent on the design rules. In this section, the design rules for the laser cutting are specified. First, the laser spot size and laser system position accuracy constraints imposed by the laser system are examined. Then, laser programmable circuit design rule considerations are made given the laser system constraints.

4.1.1 Laser System Considerations

The laser system design of Chapter 3 provides a laser spot size of 10 μm and location accuracy of 0.5 μm. Therefore, a cutting location should not be any closer than 5.5 μm from any other circuit feature when not using an aperture. Five microns comes from the radius from the cutting location and the one-half micron is the possible error. If an aperture where used, then only 2.5 μm spacing is necessary. Two microns from the center of the conductor and one-half micron for the error. If 10 μm spot size and 4 μm linewidths are used, then 3 μm of error could be tolerated provided other circuit features are further that 6 μm from the cutting location. This extra margin for error allows at least 6 movements before realignment is necessary. Figure 4-1 illustrates these limitations.

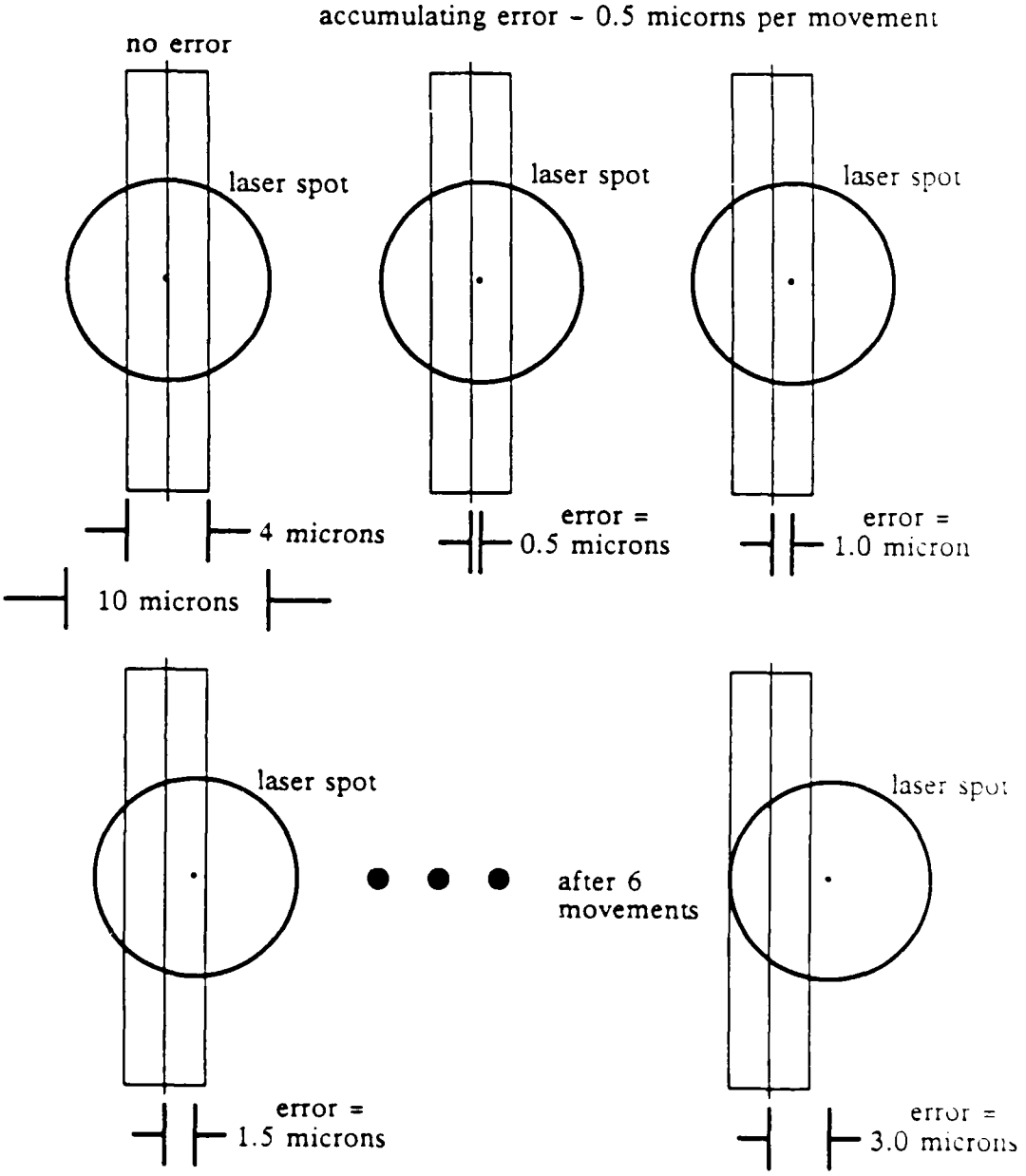


Figure 4-1 Laser Cutting Design Rules

The number of programming movements could be improved by increasing the laser spot size. Cutting a 4μm line with a 10 μm circular spot permits 6 programming move-

ments before realignment may be necessary. The spacing between the cutting location and other circuit features should be at least 8 μm in the direction of movement and 5 μm in the other direction. If an aperture was available, a 10μm long and 4μm wide rectangular spot would also permit 6 programming steps. The spacing between the cutting location and other features should be at least 8 μm in the direction of movement but only 2 μm in the other direction. The laser cutting design rule could be reduced by using rectangular spots instead of circular spots when error correction is not used.

Given that there is no aperture, the laser system requires at least 10.5 μm from the laser cutting location to any other circuit feature for a single cut and 16 μm for multiple cuts. However, 50 μm is specified as an extra margin of safety and to allow experimentation aimed at maximizing the number of programming steps. Since a 50 μm square is a considerable amount of area for 3.0 μm technologies, the layout of laser programmable circuits requires special considerations.

4.1.2 Circuit Design Considerations

Not only is it necessary to protect the circuit from the laser cuts, but it is also necessary to effectively design around the laser cutting area. Routing to the cutting area should be minimized to conserve space but note that no time delay is caused by the line to be cut. The logic value on the line is static. Thus, it is important to keep highly interconnected sections of the circuit intact and to locate the laser cutting areas adjacent to these sections.

The position of the laser cutting areas also affects automation. The location for each laser cut must be input into the controller. The task of specifying the location for each laser cut is easier when the cutting areas are arranged with regularity. When the

laser cutting areas are on a line parallel to one of the axes, one of the coordinates is constant. The other coordinate may be specified by adding a multiple of the spacing to the previous value. Minimizing the number of lines is also an effective automation aid.

Therefore, two subjective design rules result from circuit layout considerations. One, keep highly interconnected sections intact to minimize routing, and two, keep the laser cutting areas in a minimum number of lines. These laser cutting design rules are implemented in the next two sections in which the design of the laser programmable circuits is described.

4.2 32-bit Comparator Chip

The laser programmable 32-bit comparator performs four address or address field detections and encodes the results of the four comparison into three output bits. This section covers the specifications, design, and VLSI design of the circuit. The laser programmable features of this circuit are the static address bits in the comparators and the selector bits in the multiplexers.

4.2.1 Specifications

The 32-bit comparator is part of a VME bus interface chip for the AFIT CAMputer board 'Fre86'. This chip replaces the 8 to 16 packages necessary to implement using off-the-shelf components. The intent is to save board area and increase speed. The comparator delay is expected to be 15 to 25 ns, but there are no specific area or power dissipation constraints. The specified three bit output of the chip, V - valid address, R1 - Reset operation, R2 - RAM operation, is covered in section 4.2.2.4.

4.2.2 Design

The circuit design of the 32-bit comparator starts with the architectural description. Next, the components noted in the architectural description are covered. The components are the comparator array, the multiplexer array, and the output logic. The design is detailed down to the gate level.

4.2.2.1 System Architecture Design. The 32-bit comparator contains a comparator array, a multiplexer array, and an output logic block as shown in Figure 4-2. The comparator array is composed of eight comparators: four in use and four spares. The multiplexer array contains four multiplexers which select the comparator outputs.

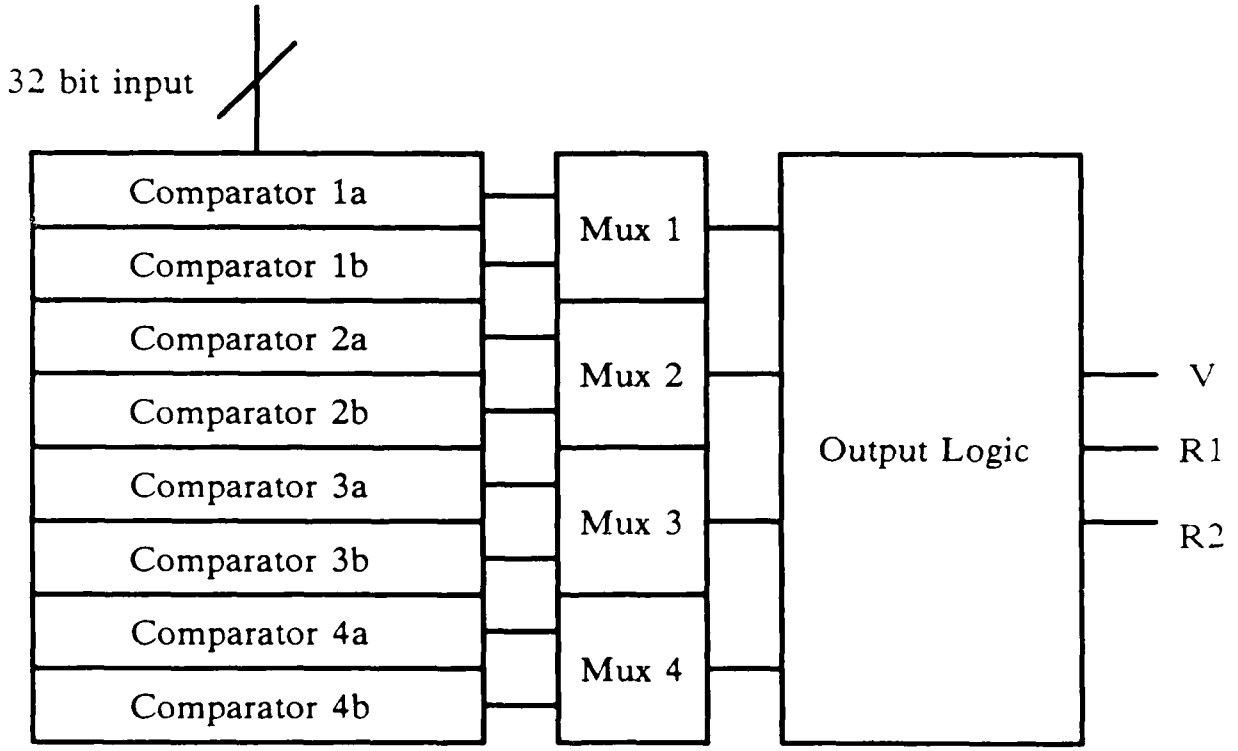


Figure 4-2 32-bit Comparator

The multiplexed comparator outputs go to the output logic block to determine the values of V, R1, R2.

4.2.2.2 Comparator Array Design. The comparator array consists of eight identical comparators which all have the same 32-bit input but detect four different addresses or address fields. Figure 4-3 is a single comparator and consists of 32 two input XOR gates and one 32 input NOR gate. The input of the XOR gates is a address bit and the other is a programmable comparison bit. The comparison is initially a logic low and may be made a logic high by cutting in location X in Figure 4-3. The output of every XOR gate may be made a don't care by cutting in location Y. The don't cares allow an address field to be detected verses a single address.

The 32 XOR outputs are all logic low when a match occurs and all the pull-down transistors are off so the pull-up transistors holds the output high. Consequently, the NOR output is a logic high and the inverter makes the comparator output a logic low for a match. When there is not a match, one or more of the 32 XOR outputs are logic high and the corresponding pull-down transistor(s) are on. Consequently, the NOR output is a logic low and the inverter makes the comparator output a logic high when there is not a match.

The critical feature of this design is the ratioing of the pull-up and pull-down transistors. The unique features of this design are the programmable bit and the don't care capability. These details of these features are covered in section 4.2.3.2. All eight of the comparators are identical and the outputs go to the multiplexer array.

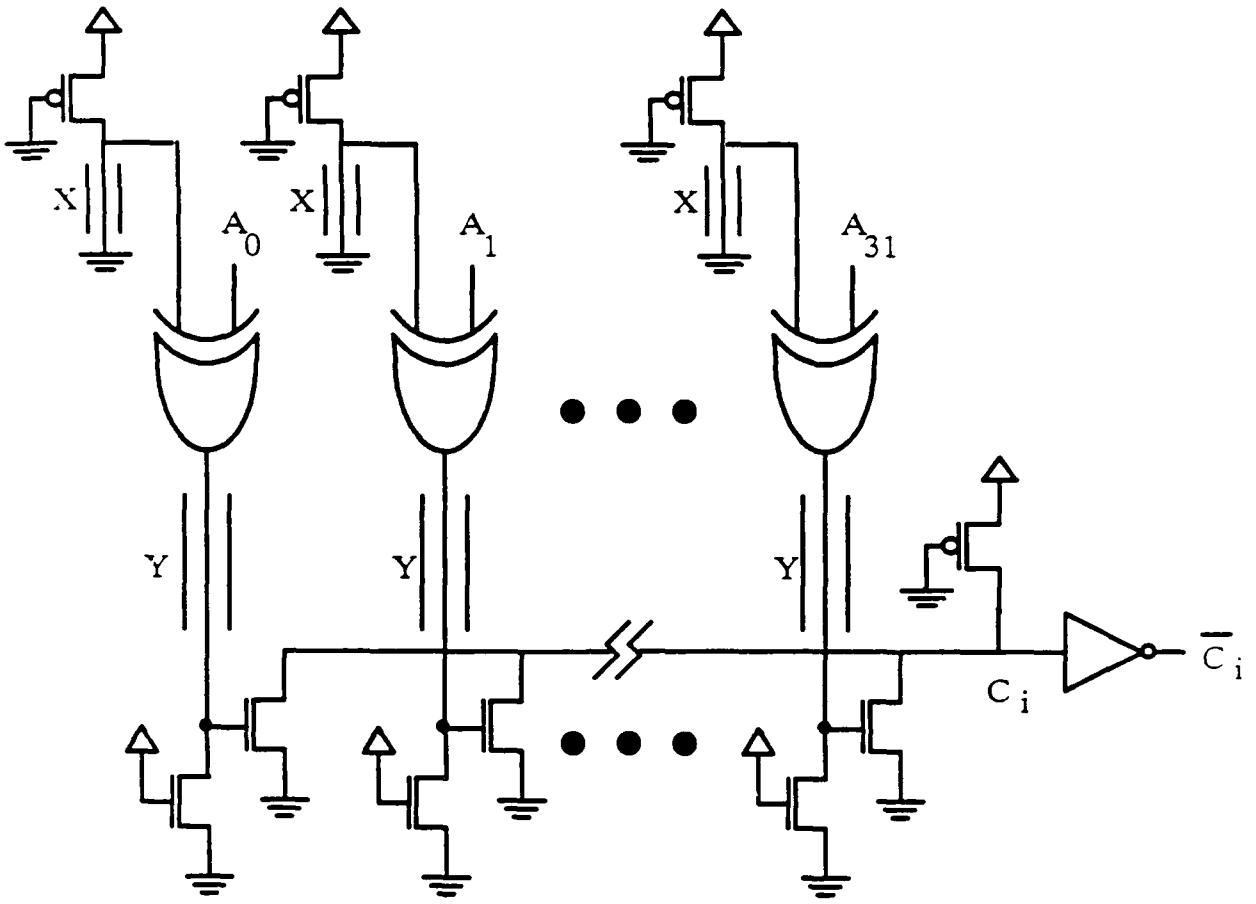


Figure 4-3 Comparator Array

4.2.2.3 Multiplexer Array Design. The multiplexer array contains four identical multiplexers. It is used to select one output from each of the four pairs of comparators. A single multiplexer is shown in Figure 4-4 and illustrates another laser programmed feature. Initially, input A is passed through because transmission gate 1 is conducting and transmission gate 2 is not. Whereas when cuts are made at locations X and Y, transmission gate 1 no longer conducts but instead transmission gate 2 conducts.

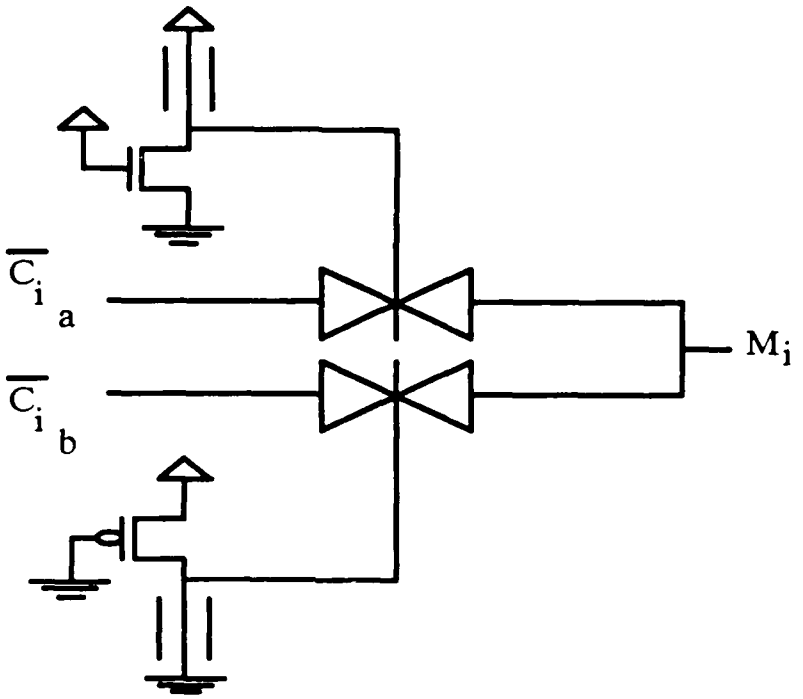


Figure 4-4 Multiplexer

4.2.2.4 Output Logic Design. Table 4-1 summarizes the logic values that are specified to be generated from the multiplexer outputs. The logic functions necessary to generate these outputs are computed in Table 4-1. The resulting output logic is shown in Figure 4-5.

4.2.3 Implementation

The layout of the chip is covered in three steps. First, the chip level layout is covered including the pad frame. Second, the layout of the programmable bits, don't care bits, and output transistor ratioing in the comparators are examined. Last, the layout of the multiplexers and the observability of the output logic is discussed.

Table 4-1. Comparator Output Signals and Boolean Logic

M1	M2	M3	M4	$\overline{V}$	$\overline{R2}$	$\overline{R1}$
0	0	0	0	1	1	1
1	X	X	X	0	1	0
0	1	X	X	0	0	1
0	0	1	X	0	0	0
0	0	0	1	0	0	0

$$V = M1 + \overline{M1}\ M2 + \overline{M1}\ \overline{M2}\ M3 + \overline{M1}\ \overline{M2}\ \overline{M3}\ M4 = M1 + M2 + M3 + M4$$

$$R2 = \overline{M1}\ M2 + \overline{M1}\ \overline{M2}\ M3 + \overline{M1}\ \overline{M2}\ \overline{M3}\ M4 = \overline{M1}\ (M2 + M3 + M4)$$

$$R1 = M1 + \overline{M1}\ \overline{M2}\ M3 + \overline{M1}\ \overline{M2}\ \overline{M3}\ M4 = M1 + \overline{M2}\ (M3 + M4)$$

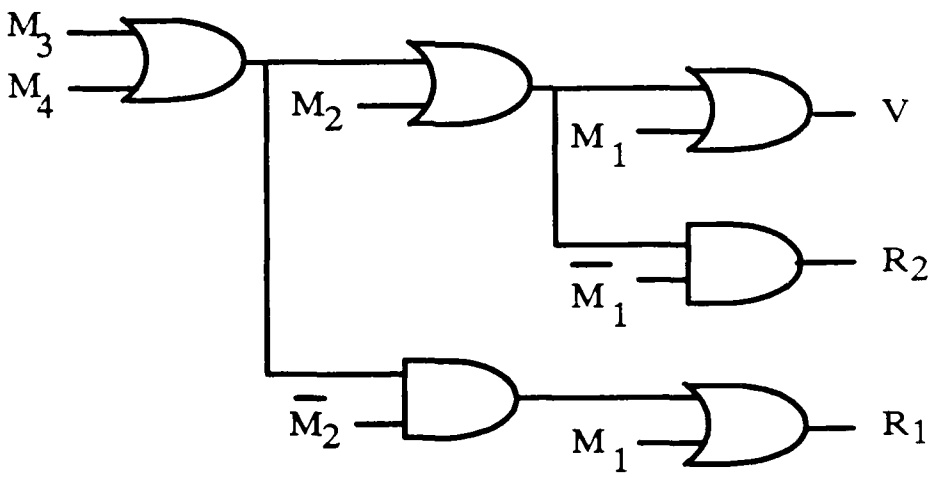


Figure 4-5 Output Logic

4.2.3.1 Chip Level Implementation. The chip is contained in a 40 pin package and is shown Figure 4-6. The pad frames and probe pads are from the AFIT VLSI cell library, but the rest of the chip is a custom layout. Observability is incorporated by the use of probe pads to get at the outputs of the multiplexers. A test structure is include in the unused area of the chip. One set of pads is connected by a metal two line and is used for the initial cutting attempts. Another set of pads have one pad connected to a metal one line to test metal one cutting.

4.2.3.2 Comparator Array Implementation. The features of interest in the comparator array are the programmable bits, the don't cares, and the output transistor ratioing. The programmable bits and don't cares are implemented as shown in Figure 4-7. Note the long and narrow gates on the programming transistors used to minimized static power dissipation. The SPICE simulations included in Appendix B show the static power dissipation for programming transistors to be approximately 20 μA. Given that there are 8 rows of 32 programming transistors, approximately 5.12 mA of current is attributable to the programming transistors. The n-type transistors at the XOR outputs contribute 85 μA for every high output, totaling 21.8 mA when all 256 outputs are high. The p-type pull-up transistors contribute 1.4 mA for each low comparator output, totaling 11.2 mA when all comparator outputs are low. The total current from all three sources should range from 5.12 mA when all the address inputs are low to 38.1 mA when all the inputs are high.

The 50 μm design rule specified in section 4.1.1 and the use of rows of programmable locations discussed in section 4.1.2 are illustrated in Figure 4-7. The loop in the line serves two useful purposes: it routs the line through the cutting location to the necessary power or ground connection, and it extends the number of programming cuts which can

Figure 4-6 32-bit Comparator Photograph

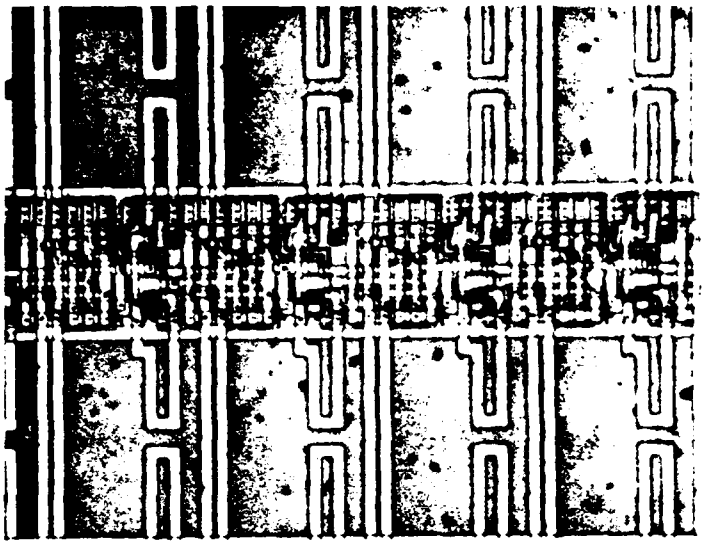


Figure 4-7 Comparator Array Photograph

be made without realignment. With a 20 μm spot size and 0.5 μm accuracy, up to 24 programming steps can be made instead of the 16 calculated in Section 4.1.1.

The output transistor ratioing is accomplished using SPICE simulations to achieve 10 ns rise and fall times. The pull-up and pull-down ratios are determined in conjunction with the output inverter sizing. The SPICE simulation for the selected output transistor sizings and the modeled circuit are included in Appendix B.

4.2.3.3 Multiplexer Array and Output Logic Implementation. The multiplexer array layout is shown in Figure 4-8. The multiplexer programming locations are on the same line as the comparator locations. The probe pads for observing the output of the multiplexers are also shown in Figure 4-8. The output of the multiplexers and the

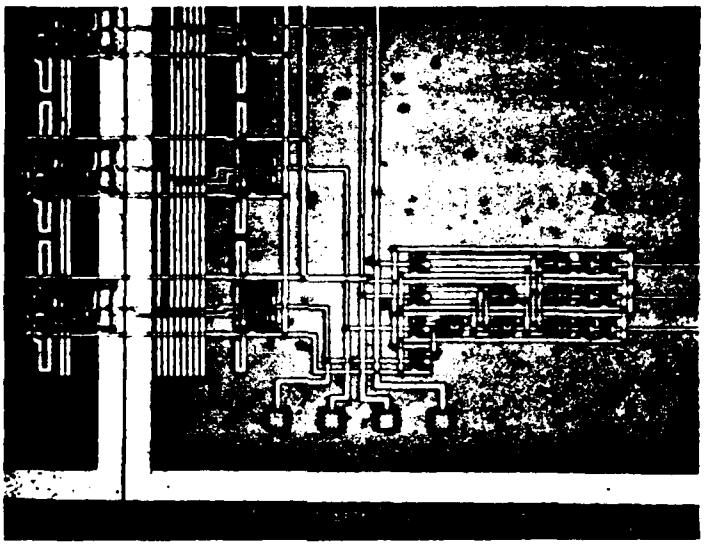


Figure 4-8 Multiplexer Array and Output Logic Photograph

inputs to the output logic can be observed at these pads.

4.3 Serial Multiplier

The laser programmable multiplier is a serial multiplier cell for use in a pipelined bit-serial multiplier. This section covers the laser programmable multiplier specification, design, and VLSI implementation. The specification includes an explanation of the algorithm this multiplier cell supports. The design presents the gate level structure of the circuit. Finally, the implementation explains the layout of the chip with special emphasis on the laser programmable feature.

4.3.1 Specifications

The laser programmable multiplier may be applied to any serial multiplier in which the multiplier is static. However, two existing applications specify a pipelined bit-serial architecture using Booth's Quaternary encoding algorithm: the Winograd Fourier Transform multiplier stage [Cou85] and a Finite Impulse Response filter multiplier stage [Lin86]. Consequently, this laser programmable multiplier cell is a Booth's Quaternary cell. The other requirement is to meet the operational speed of the 50 to 70 MHz clock. This results in a 14 to 20 ns delay through the delay cells in the laser programmable multiplier cell. No area or power dissipation specifications exist, but the realization that several cells are necessary for the serial multiplier, emphasizes the need for minimum area and power dissipation. The WFT 16-point multiplier stage performs 36 26-bit multiplications in parallel. Therefore, laser programmable multiplier cells are required to perform the multiplication when Booth's Quaternary encoding is employed.

Booth's Quaternary encoding is an algorithm in which N multiplier bits are encoded into N/2 bits [Boo51]. The benefit of this algorithm is that it needs only one half the number of multiplier cells as non-encoded multiplication algorithms. The side effect of this algorithm is that only N/2 most significants bits of accuracy results verses N bits of accuracy for other algorithms. The encoded bits take on one of five values: +2, +1, 0, -1, -2. The value of any one encoded bit is determined by three of the unencoded bits. The encoding rules are shown in Table 4-2.

To illustrate how Booth's Quaternary encoding is performed consider the following example of encoding the 15 bit 2's complement number 100011101001100. First, place a zero to the right of the least significant

Table 4-2. Booth's Quaternary Encoding Rules

b_{i+1}	b_i	b_{i-1}	eb
0	0	0	0
0	0	1	+1
0	1	0	+1
0	1	1	+2
1	0	0	−2
1	0	1	−1
1	1	0	−1
1	1	1	0

i = 0, 2, 4, ...

1|1000111010011000| = -14516 decimal

110 000 011 110 010 001 110 000
--- --- --- --- --- --- --- ---
-1 0 +2 -1 +1 +1 -1 0

$(-1)2^{14} + (0)2^{12} + (+2)2^{10} + (-1)2^{8} + (+1)2^{6} + (+1)2^{4} + (-1)2^{2} + (0)2^{0} = -14516$ decimal

bit. Second, encode in groups of three bits, shifting the three bit field two places to the left after each encoding. Since there are only two bits in the last group, sign extend one bit to complete the three bit field. When there are an even number of bits, no sign extension is necessary. The eight encoded serial multiplications perform the same multiplication as 15 nonencoded serial multipliers. The design of the arithmetic logic to perform the five distinct multiplications is covered next.

4.3.2 Design

The design of the laser programmable circuit starts with the system architecture description. The description of the circuit architecture decomposes the system into three

parts: *the input signal generator, the selector, and the arithmetic logic*. *The selector is* the laser programmable feature that allows selection of the encoded coefficient.

4.3.2.1 System Architecture Design. A block diagram of the laser programmable multiplier cell is shown in Figure 4-9. The components of the multiplier cell are the input signal generator, the selector, and the arithmetic logic. The input signal generator provides the five versions of each input signal necessitated by Booth's Quaternary encoding and the selector chooses one of the values to control the arithmetic section. The arithmetic section generates the five partial products and carries and the selector chooses the result. The selected partial product is latched for input into the next serial multiplier cell. The selected carry is latched for input during the next clock cycle.

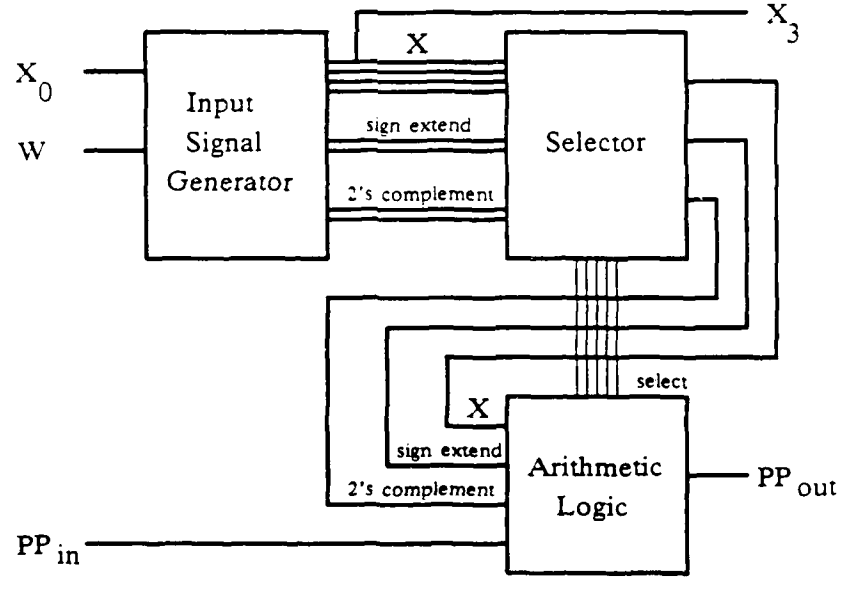


Figure 4-9 Laser Programmable Serial Multiplier

4.3.2.2 Input Signal Generator Design. The inputs are the serial streams of multiplier terms, partial product from the previous stage, and the word control signal. Depending on the coefficient of the multiplier, master slave flip-flops are used to provide one, two, or three delays before output to the selector section as shown in Figure 4-10. For example, the multiplier terms are delayed two times for multiplication by +1 and -1, and three times for multiplication by +2 or -2 to line up with the partial product of the previous stage. The partial product terms are not delayed by the input signal generator. The word control signal keeps step with the multiplier term. The word control signal is used to to perform 2's complementing at the beginning of a word and sign extend at the end of a word if the multiplier is negative.

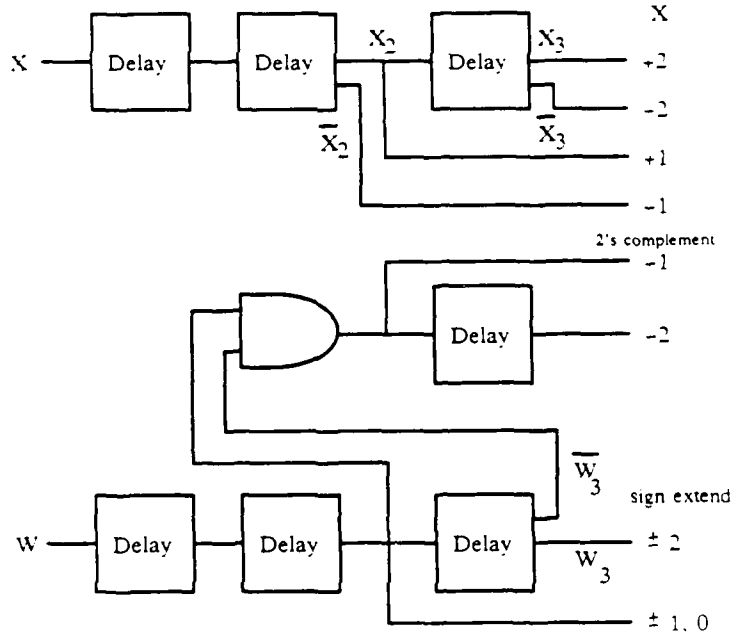


Figure 4-10 Input Signal Generator

4.3.2.3 Selector Design. The selector is the laser programmable feature of the laser programmable multiplier. The selector chooses the correctly timed multiplier term, 2's complement control, and sign extend control depending on the coefficient of the multiplier. Figure 4-11 shows how programmable multiplexers are used to select the desired signals. The output of the three multiplexers is selected by cutting the line corresponding the desired coefficient. Initially, no output is selected so that a cut of a programmable location selects the corresponding coefficient.

4.3.2.4 Arithmetic Logic Design. The arithmetic logic section generates the five possible partial products and carries. Table 4-3 lists the Boolean logic implemented in the arithmetic logic. The selector chooses the results latched into the partial product and carry delay cells. An extra carry bit is used because of the possibility of multiplication by +2 or -2. The block diagram of the arithmetic logic is shown in Figure 4-12. Sign extend and 2's complement is also performed in this section. The sign extend and 2's complement signals are created in the input signal generator. The signal choice occurs in the selector according to the multiplier value and sign. A high sign extend signal latches the previous partial product into the partial product delay cell. A high 2's complement latches a one into the carry delay cell.

The logic gate implementation for the partial product and the carry is shown in Figure 4-13. Since the Boolean logic overlaps, the gate level implementation may be minimized considerably. The effect of the minimization is shown in Figure 4-14. Signals from the selector are decoded and used to select the proper output of both the partial product and the carry. The selected outputs go to delay cells.

Table 4-3. Partial Product and Carry Boolean Logic

Multiplier	0	0		
(M)ultiplicand	X	X		
Product	0	0		
PP_{in}	0	1		
$CY0_{in}$	0	0		
PP_{out}	0	1	=	PP_{in}
$CY0_{out}$	0	0	=	0
$CY1_{out}$	0	0	=	0

Multiplier	1	1	1	1	1	1	1	1		
(M)ultiplicand	0	0	1	1	0	0	1	1		
Product	0	0	1	1	0	0	1	1		
PP_{in}	0	1	0	1	0	1	0	1		
$CY0_{in}$	0	0	0	0	1	1	1	1		
PP_{out}	0	1	1	0	1	0	0	1	=	$M \bullet PP_{in} \oplus CY0_{in}$
$CY0_{out}$	0	0	0	1	0	1	1	1	=	$M\,CY0_{in} + M\,PP_{in} + PP_{in}\,CY0_{in}$
$CY1_{out}$	0	0	0	0	0	0	0	0	=	0

Multiplier	2	2	2	2	2	2	2	2	2	2	2	2
(M)ultiplicand	0	0	1	1	0	0	1	1	0	0	1	1
Product	0	0	2	2	0	0	2	2	0	0	2	2
PP_{in}	0	1	0	1	0	1	0	1	0	1	0	1
$CY0_{in}$	0	0	0	0	1	1	1	1	0	0	0	0
$CY1_{in}$	0	0	0	0	0	0	0	0	1	1	1	1
PP_{out}	0	1	0	1	1	0	1	0	0	1	0	1
$CY0_{out}$	0	0	1	1	0	1	1	0	1	1	0	0
$CY1_{out}$	0	0	0	0	0	0	0	1	0	0	1	1

$$PP_{out} = PP_{in} \bullet CY0_{in}$$

$$CY0_{out} = \overline{M}\,CY1_{in} + \overline{M}\,PP_{in}CY0_{in} + M\,\overline{CY0}_{in}\,\overline{CY1}_{in} + M\,\overline{PP}_{in}\,\overline{CY1}_{in}$$

$$CY1_{out} = M\,PP_{in}CY0_{in}\,\overline{CY1}_{in} + M\,\overline{CY0}_{in}CY1_{in}$$

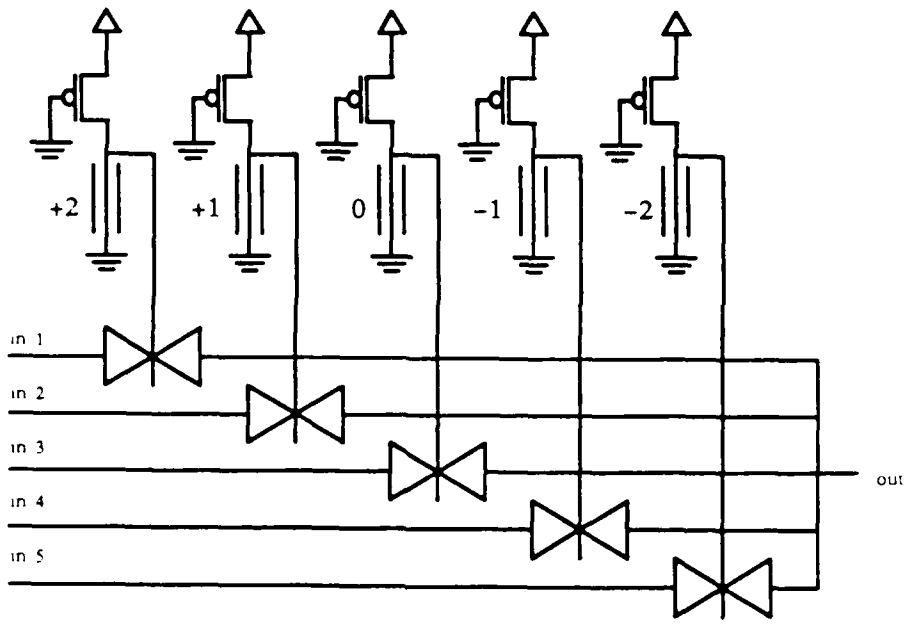


Figure 4-11 Selector

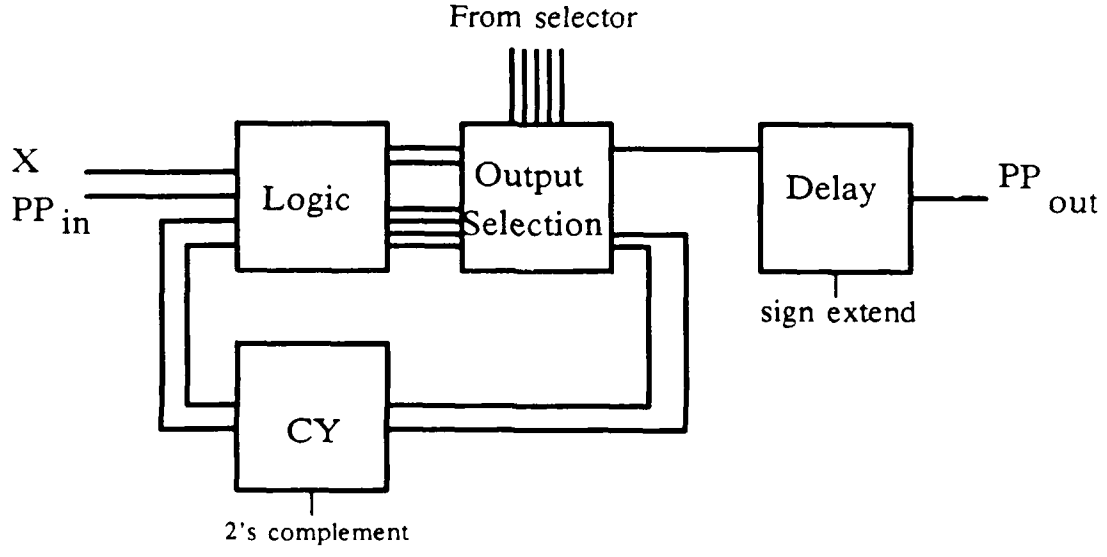


Figure 4-12 Arithmetic Logic Block

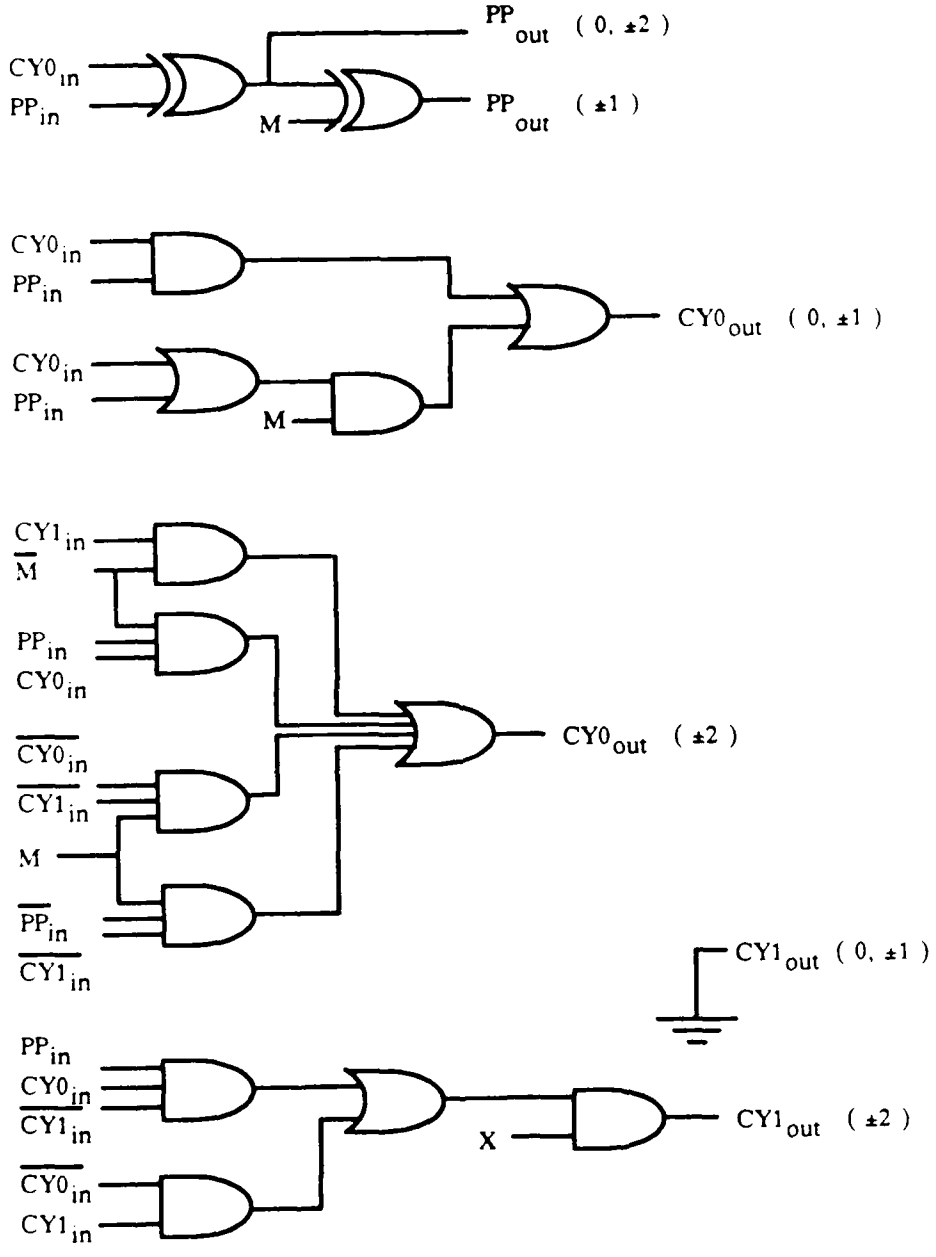


Figure 4-13 Gate Level Arithmetic Logic Diagram

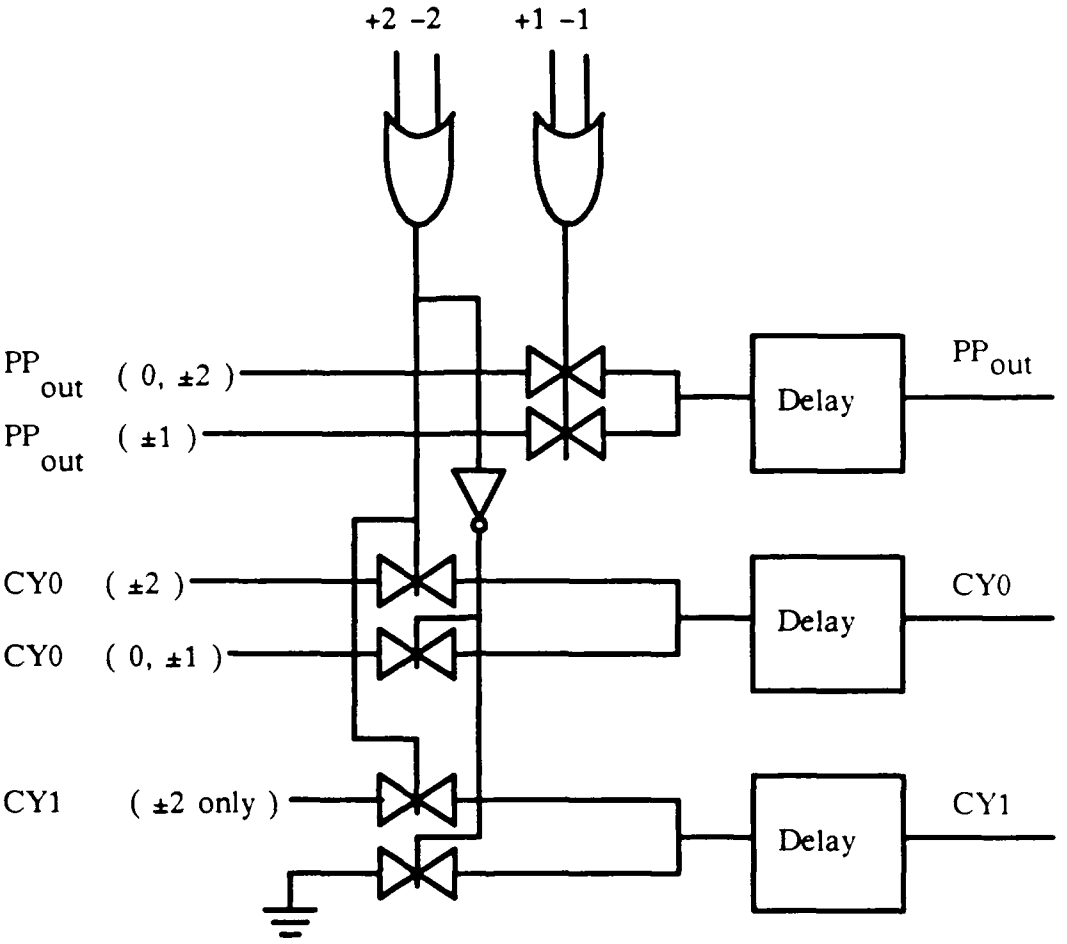


Figure 4-14 Arithmetic Logic Output Selection

4.3.3 Implementation

The features of interest in the implementation of the laser programmable multiplier are the selector and testing considerations. Therefore, the layout of the chip is covered in three steps. First, the chip level layout is examined. Then, the layout of the selector, the laser programmable feature of the chip, is discussed. Finally, observability for testing the chip is considered.

4.3.3.1 Chip Level Implementation. The Cifplot of the chip is Figure 4-15 The pads are from the AFIT VLSI cell library, but the rest of the chip is a custom layout. To aid testing, the laser programmable lines are brought off chip through a pad. This allows functional testing before laser programming. In addition, there are enough pins provide sufficient observability without probe pads.

4.3.3.2 Selector Design Implementation. The laser programmable feature of the selector are the three multiplexers. The multiplexers contain five transmission gates which are all not transmitting initially. Disconnecting any one of the five programming lines from ground turns the corresponding transmission gates on and the corresponding inputs are selected. Location A in Figure 4-15 is the laser cutting area. The programming lines run vertically to all three multiplexers such that three transmission gates, one for each multiplexer, conduct when a disconnection is made. The programming lines would normally connect to the ground line at location A but for this prototype the lines run to pads for testing

4.3.3.3 Implementation of Observability. The location selected to provide observability for testing is designated as location B in Figure 4-15. This location is between the selector and the arithmetic logic block. The output of the input signal generator and selector can be tested, and the input to the arithmetic logic block can be determined The observable signals are the 2's complement and sign extend control, and the two carry bits

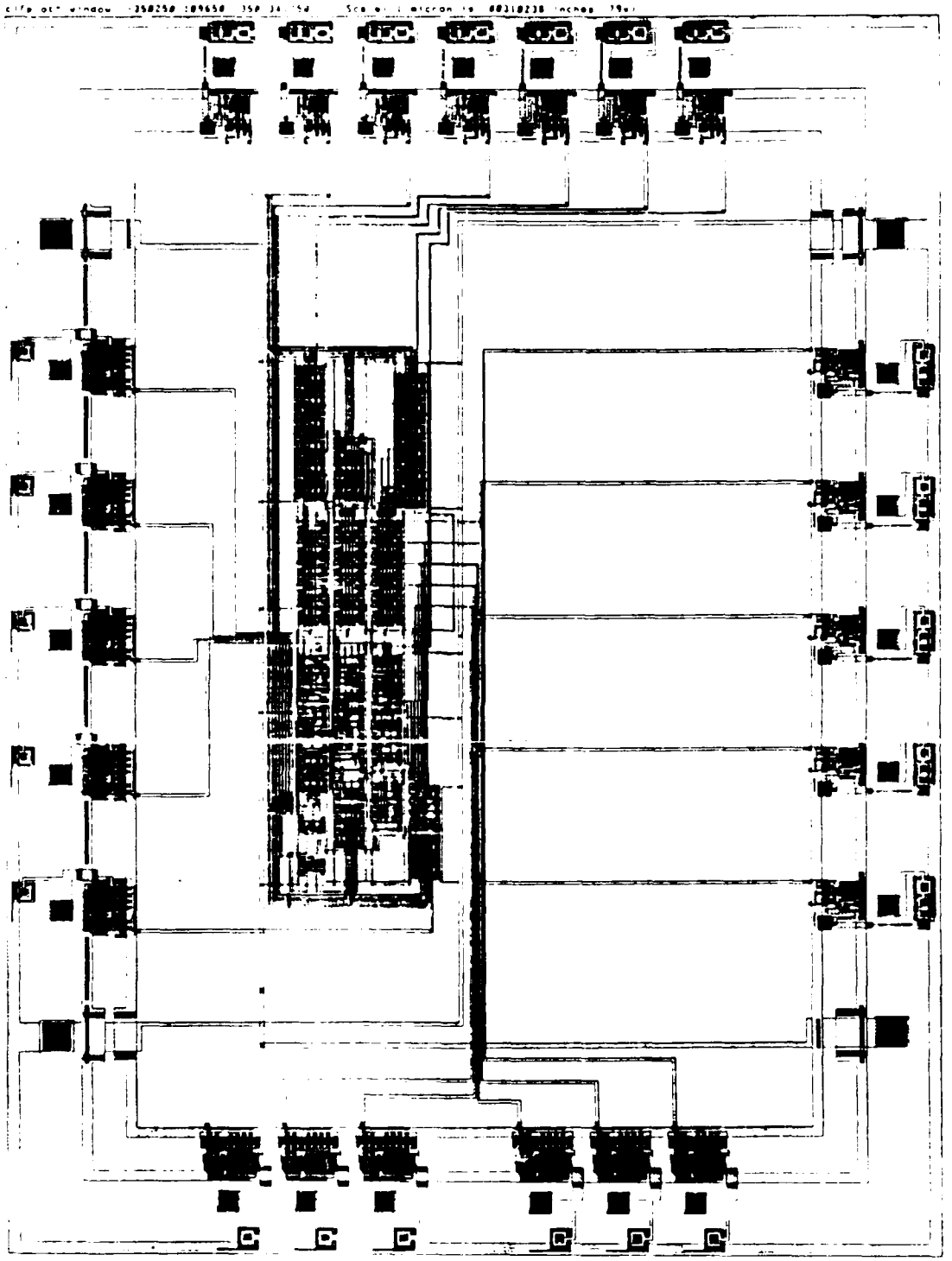


Figure 4-15 Laser Programmable Serial Multiplier Cifplot

CHAPTER 5

Results

5.1 Introduction

This chapter contains the results of the laser system, design rule, and the laser programmable comparator tests. The laser system testing included laser alignment, laser cutting, and controller accuracy. The design rules testing involved evaluating the effect of the controller accuracy and laser cutting effects on the circuit layout. Power-up, functional, and speed tests were conducted on the laser programmable comparator. The laser programmable serial multiplier test results are not included because it did not return from fabrication in time.

5.2 Laser System

The laser system testing had two phases: phase one tested the laser and optical arrangement, and phase two tested the automation of the laser system. Phase one of the tests included the following milestones:

a. merging the beams,
b. cutting with a beam before the microscope optics,
c. aiming the beams into the microscope, and
d. cutting with the beam after the microscope optics.

Phase two had the milestones of determining:

a. single step accuracy,
b. multiple step accuracy, and
c. remote programming capability.

This section covers the evaluation of the laser system and the revisions made to meet the

objectives of the research.

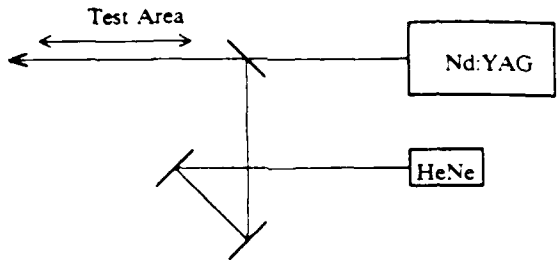
5.2.1 Evaluation of Laser System

5.2.1.1 Merging the Beams. The initial arrangement of the lasers and optics demonstrated the capability to merge the beams. Figure 5-1a shows the arrangement in which was merged with the visible laser. The visible laser was steerable with four degrees of freedom and merged with the cutting laser using a beam splitter. The coincidence of the lasers the beams was accomplished at distances ranging from 0.1 m to 3 m from the beam splitter.

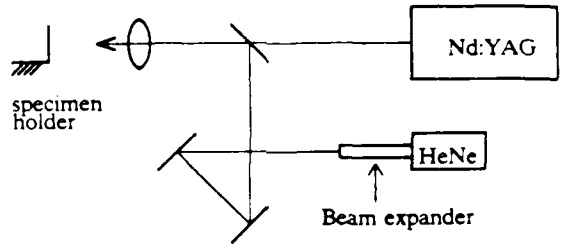
The visible laser was easily steered and merged with the cutting laser. The coincidence was verified by detecting the cutting laser with photosensitive material and seeing if the spots lined up from the near to far field. Next, cutting was attempted by focusing the beams to maximize power density. This was done by putting a lens in the path of the lasers.

5.2.1.2 Cutting Before the Microscope Optics. The focal length of the lens was 10 cm and it magnified the cutting beam to a 20 μm spot size. The 32-bit comparator chip was mounted on an adjustable stage which was used to place the test structure into the path of the beams as in Figure 5-1(b). The line to be cut was placed in the path of the beams and then was cut by one pulse of the cutting laser. The result was the cut shown in Figure 5-2

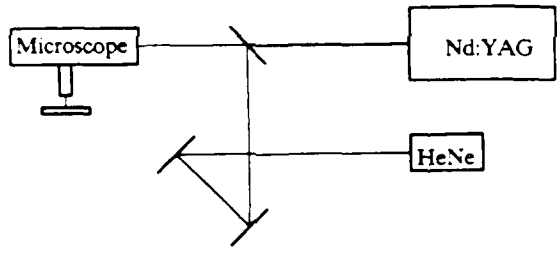
5.2.1.3 Aiming the Beams into the Microscope. Aiming the beams into the microscope was similar to merging the beams. An additional consideration was the location of the microscope port and the relative angle of the beam. The location of the port



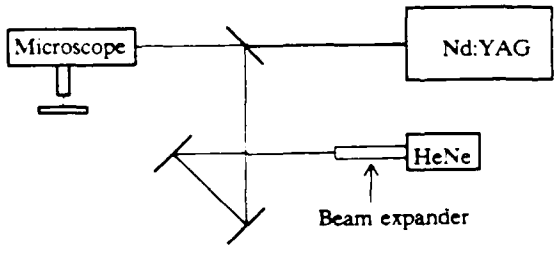
(a) Beam Merging Arrangement



(b) Cutting Arrangement
(20 um cut achieved)



(c) Beams Injected into Microscope



(d) Beam Expander Added for Visual Laser
(10 um spot size achieved)

Figure 5-1 Laser and Laser Optics Testing Arrangements

Figure 5-2 Laser Cut

was adjusted until the visible laser light was seen coming out of the microscope objective. However, the test revealed that the visible beam was too large to be focused to a spot by the microscope optics spot. Consequently, the beam expander was needed to reduce the diameter of the beam before the microscope as shown in Figure 5-1c.

5.2.1.4 Cutting After the Microscope Optics. The system set-up for cutting after the microscope optics is shown in Figure 5-1d. The lack of a beam expander and mirrors for the Nd:YAG laser prevented cutting after the microscope optics. The beam expander for the visible laser was adjusted until the beam coming out of the microscope objective was minimized. Using the features of the test structures as a reference, the spot size of the visible laser was estimated to be 10 μm in diameter. The output of the

microscope objective was monitored using photosensitive paper. The Nd:YAG pulse was detected on the photosensitive paper, but the power density was not high enough for cutting. Cutting was not accomplished, but focusing a laser through the microscope optics was demonstrated.

A lens was used just outside the microscope port to focus the laser. The focal length was too short to focus the beam at the microscope output. The beam came to a focus at a mirror and burned spots the size of pinheads on the mirror. Fortunately, the mirror only reflects the illuminating light and not the image. This result demonstrated that a beam expander is needed to achieve the desired beam size.

5.2.1.5 Single Step Accuracy. The laser programmable comparator chip was used as a reference to make the measurements to determine the single step accuracy. Movements from one programming location to another served as the test. The distances were 124.5 μm in the y direction and 265.5 μm in the x direction. The results the tests are shown in Table 5-1.

The first movement of a program was found to have a considerable error due to slack in the belts. The slack occurs between the motors and the fine adjustment mechanism, and inside the microprobe station. After the first movement, the error in each following movement reduced to a reasonable value. Therefore, the magnitude of the programmed value for the first two movements was changed to compensate for the error in the first movement. Another way to compensate for the large error for the first movement was to use the return to home feature of the controller. The first move was followed by a return to home which ends up within - 2 μm from the starting position.

Table 5-1. Single Step Accuracy

Single Step Accuracy in microns

Without correction				With correction			
First Step	Following Steps 2	3	4	First Step	Following Steps 2	3	4
24	+1	−1	+2	−2	+2	+1	−1
25	−2	−1	+1	−2	−1	+1	+1
25	+1	−1	+1	−1	+1	+2	−1
26	−1	+2	−1	−2	+2	+1	+1
24	−1	0	+2	0	+2	+2	−2

The error associated with the controller and motor was estimated in Chapter 3 to be 0.5 μm for each movement. These tests show that more error exists. However, the tests revealed that the random nature of the error allows a considerable number of movements to occur before the error accumulates to a unacceptable level. The tension in the belts attached to the motors was increased to minimize the slack. The tension in the belts inside the microprobe station was not adjustable. The manufacturer, MicroManipulator, redesigned the drive system when they built an automated version of the microprobe station because of slack in the belts.

5.2.1.6 Multiple Step Accuracy. To determine multiple step accuracy, two series of movements were programmed into the controller. One program was a series of 31 movements and the other program was a series of 7 movements. The results of the tests are shown in Table 5-2.

Table 5-2. Multiple Step Accuracy

Multiple Step Accuracy

Test	Maximum Number of Steps Before 16 microns of Accumulated Error
1	27
2	31 Max
3	31 Max
4	29
5	31 Max

Error was estimated to accumulate at the rate of 0.5 μm for each movement. Consequently, the maximum number of movements, for cutting a 4 μm wide line with a 20 μm spot, was estimated to be 16. By adding the loop in the line to be cut the maximum number of movements was shown to be 24. Despite not attaining the 0.5 μm accuracy for a single movement, the multiple step accuracy achieved the 24 step accuracy estimate. The random nature of the actual single step error improved the multiple step accuracy. The minimum number of movements before unacceptable error occurred was 27 for two tests and for the other three tests all 31 movements were able to be made without unacceptable error accumulation.

5.2.1.7 Remote Programming Capability. The controller worked well in the remote mode. The commands worked the same way as when entered directly. The software written to operate the controller in the remote mode supported all the com-

mands properly. In addition, the reading and writing of the controller's memory to and from files on the remote computer worked effectively. The programs written to generate the program for the comparator also worked effectively. The code written for the user interface and comparator programming are included in Appendix A.

5.3 Design Rules

The 50 μm design rule depended on the cutting beam spot size and the accuracy of the controller. Since the 10 μm laser spot size and a 0.5 μm controller accuracy were achieved, the design rule was valid. In addition, the laser programming had no effect on nearby circuit features. Consequently, the design rule could be changed to be a function of spot size and accumulated error plus 1 or 2 μm instead of the 30 μm specified in section 4.1. The major design rule factor is the laser spot size and error accumulation combination. Cutting can be done 1 or 2 μm from other features, but to achieve long series of movements without realignment, larger spot sizes are required.

5.4 Laser Programmable Circuit Testing

This section is a description of the testing of the laser programmable 32-bit comparator. Testing of the laser programmable serial multiplier is not covered because of delay in fabrication. Testing the 32-bit comparator involves the power-up, pre- and post-programming functional tests, and pre- and post-programming speed tests.

5.4.1 32-bit Comparator Power-Up Tests

The 32-bit comparator has 32 x 16 + 8 programming transistors contributing to static power dissipation. Figure 5-3 shows the power-up curve for an unprogrammed chip in five different configurations. In the first configuration, all address bits were tied to ground. For each successive test, one address bit was tied to power. Each high

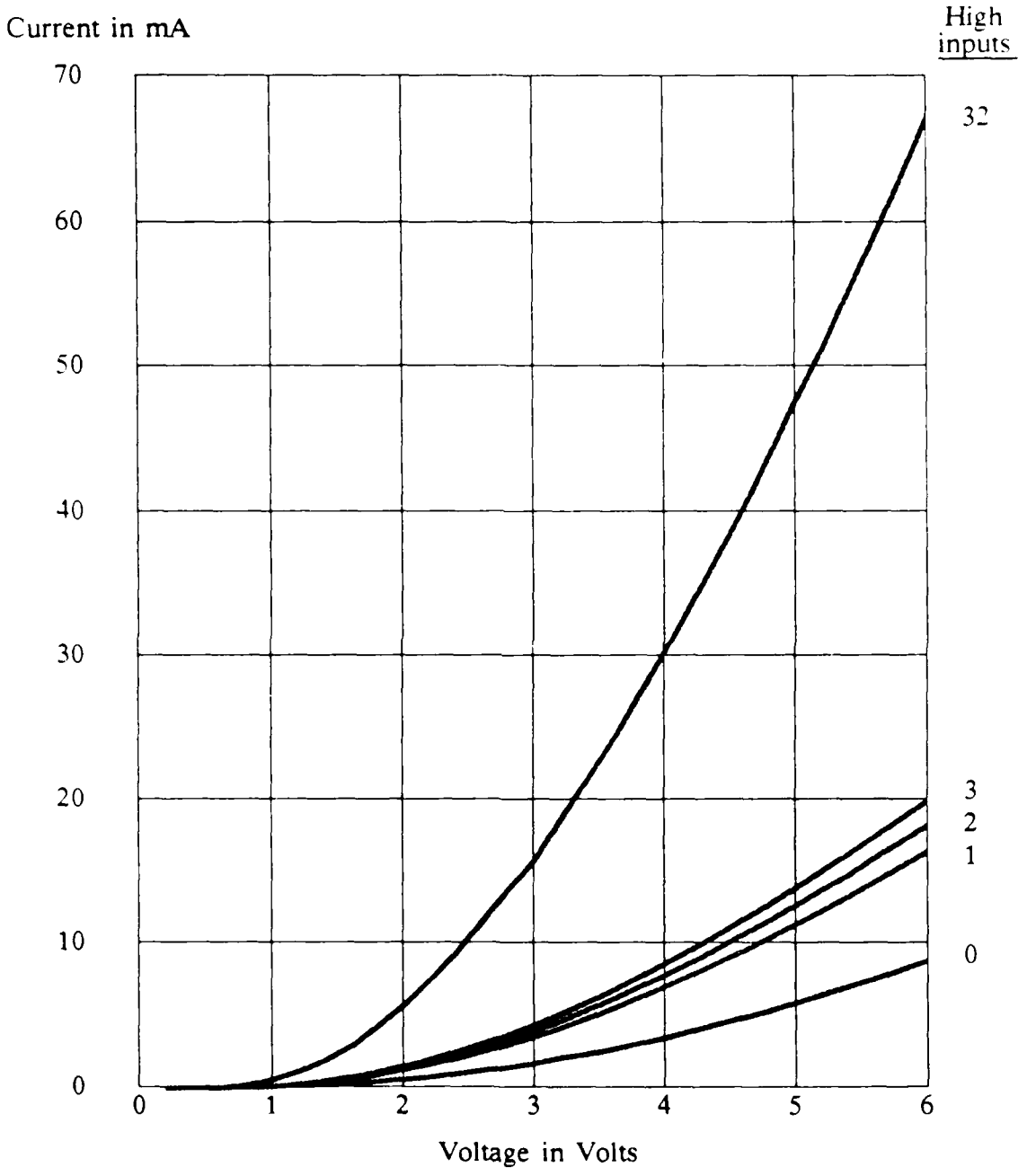


Figure 5-3 Laser Programmable Comparator Chip Power-Up

address bit caused a logic high XOR output in the comparators. Every logic high XOR output turned on one transistor in the 32-input NOR gate. Therefore, the supply current increased linearly for each high address bit.

In Section 4.2.3.2, the static current attributable to these transistors was estimated to range between 5.1 and 38.1 mA depending on how many address bits were high. The power-up tests resulted in a range of current from 5.91 to 47.98 mA. The test results show that the chip powered-up as expected.

5.4.2 32-bit Comparator Functional Tests

The output of the chip was specified in Table 5-4. One unprogrammed and four programmed chips were functionally tested. All five chips produced logically correct outputs with the voltage levels shown in Table 5-4. The tests were performed staticly by connecting the appropriate address bits to power or ground. Then the output voltage levels were measured.

5.4.3 32-bit Comparator Speed Tests

The speed tests measured the delay from the input to the output. Initial tests using a coaxial cable lead with standard clips on the input and a high impedance probe on the output resulted in delays on the order of 70 ns. The plot of the output and a reference signal, the inverse of the input, is shown in Figure 5-4(a). The input waveform was too noisy so the test was reaccomplished with the input probe modified to minimize capacitance. To match the impedance of the input cable to the load, a 50 ohm resistor was connected between it and ground. The result was a slightly less noisy input waveform and a delay on the order of 50 ns. The second set of waveforms are included as Figure 5-4(b). The tabular results of the speed tests are in Table 5-4

Table 5-3. 32-bit Comparator Functional Tests

Result of comparison				Voltage of output		
C1	C2	C3	C4	$\overline{V}$	$\overline{R2}$	$\overline{R1}$
N	N	N	N	4.92	4.92	4.92
Y	X	X	X	0.00	4.93	0.00
N	Y	X	X	0.00	0.00	4.92
N	N	Y	X	0.00	0.00	0.00
N	N	N	Y	0.00	0.00	0.00

Y - Detection

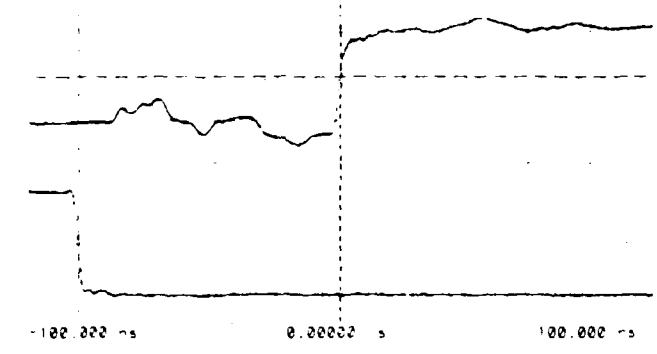
N - No Detection

X - Don't Care

Table 5-4. 32-bit Comparator Speed Tests

Output	Delay (ns)	
	Rising edge	Falling edge
V	52.78	35.44
R2	47.81	57.00
R1	51.02	36.99

(a)

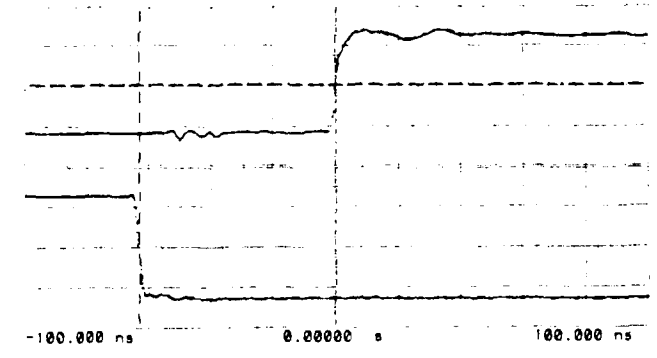


-100.000 ns 0.00000 s 100.000 ns

Ch. 1	= 2.000 volts/div	Offset	= -100.0 mvolts
Ch. 2	= 1.000 volts/div	Offset	= 1.190 volts
Timebase	= 20.0 ns/div	Delay	= 0.00000 s
Delta T	= 84.800 ns		
Start	= -84.400 ns	Stop	= 400 ps
Delta V	= 0.000 volts		
Vmarker1	= 120.0 mvolts	Vmarker2	= 120.0 mvolts

Trigger mode : Edge
On Pos. Edge on Ch1
Trigger Levels
Ch1 = -100.0 mvolts
Holdoff = 70.000 ns

(b)



-100.000 ns 0.00000 s 100.000 ns

Ch. 1	= 2.000 volts/div	Offset	= 40.00 mvolts
Ch. 2	= 1.000 volts/div	Offset	= 1.190 volts
Timebase	= 20.0 ns/div	Delay	= 0.00000 s
Delta T	= 63.200 ns		
Start	= -63.200 ns	Stop	= 0.00000 s
Delta V	= 0.000 volts		
Vmarker1	= 99.99 mvolts	Vmarker2	= 99.99 mvolts

Trigger mode : Edge
On Pos. Edge on Ch1
Trigger Levels
Ch1 = 40.00 mvolts
Holdoff = 70.000 ns

Figure 5-4 Laser Programmable Comparator Delay

CHAPTER 6

Conclusions and Recommendations

6.1 Conclusions

The objective of this research was to develop a laser programming system and demonstrate the system on a programmable circuit. The laser programming system and two laser programmable circuits were designed, implemented, and tested during this thesis effort. Laser alignment, laser cutting, and x-y stage automation were demonstrated during the research. However, the system could not be integrated because of the lack of some equipment. The laser programmable comparator chip functioned properly, but had a larger delay than desired. This section covers these conclusions in more detail.

6.1.1 Laser Programming System

The laser programming system was an overall success. The cutting of conducting paths with a laser was demonstrated. The automation components were shown to work effectively. However, the lack of a beam expander to condition the beam and the mirrors to steer the beam prevented full-scale implementation of the laser system. Another contributing factor was that the microscope optics were not designed for use at the wavelength and power density of the Nd:YAG laser.

6.1.2 Laser Programming System Limitations

Working with the laser programming system was limited by two factors. First, the lasers were located separately from the rest of the components. Thus, to assemble the system either the lasers or the other components needed to be moved. Constant moving of this equipment is not recommended because of the sensitivity of the cutting laser and the microprobe station. Second,

lack of optics for the cutting laser limited the usefulness of the system. The optics were dedicated to other research projects or had been destroyed by previous users.

A rectangular aperture for the cutting beam and the aiming light would improve the laser system. The aperture would allow spot sizing without changing the power density. The incident beam size would cover the largest opening of the aperture. Then, the aperture opening could be adjusted to obtain the desired spot size.

6.1.3 Design Rules

The design rules specified for the layout of the laser programmable circuits provided sufficient clearance from other circuit features. In fact, an improvement in the design rule occurred as a result of the testing of the laser system. Laser cutting was selective and was accomplished with no affect on adjacent circuit features. Locating the cutting locations in a row or column was found to aid automation.

6.1.4 Laser Programmable Circuits

In addition to constructing the laser system, two laser programmable circuits were designed and one was tested as a result of this research. The 32-bit comparator functioned properly and served as a laser programming demonstration chip. The serial multiplier serves as a source for further study of laser programming. Next, the recommendations which result from the conclusions of this research are presented.

6.2 Recommendations

The first four recommendations are improvements to the laser programming system. The first two recommendations results from the conclusion that the laser programming system was limited by the laser and microscope optics. The third comes from the conclusion that an aperture is needed for the laser system. The fourth is an alternative to acquiring the individual pieces to construct the laser system. The last two recommendations concern a system enhancement and

further study of laser programming applications.

6.2.1 Optics for Microscope

The first recommendation is that a set of microscope optics rated for use at the wavelength and power level of the cutting laser be acquired. The manufacturer of the MicroZoom microscope will have such a set of optics commercially available in January 1988. The estimated cost of the replacement set is $1000 and the optics can be installed locally by the user.

6.2.2 Laser and Optics

The limitations associated with the assembly of the laser programming system could be avoided by acquiring a dedicated laser and laser optics and placing all of the components in a permanent location. As a minimum, a dedicated set of mirrors, beam splitters, and beam expanders should be acquired if this research is to continue. This would avoid the limitations associated with sharing much used sensitive equipment. A beam splitter costs $1000, and the beam splitters and mirrors are $100 each.

6.2.3 Aperture for Cutting Laser

The third recommendation is to acquire an aperture for the cutting laser beam. As stated in Chapter 2, an aperture allows changing the spot size without changing the beam's power density. The cutting laser beam's power density could be set at an optimum level, then different size cuts could be made by adjusting the aperture. The cost of an aperture is $500.

6.2.4 Acquisition of a Laser Programming System. The possibility of acquiring a commercial laser programming system should also be considered. Commercial systems are available which can perform all the tasks required of the system developed during this research. In addition, other laser processes such as writing and connecting conducting paths on integrated circuits can be done with one of these systems. The system would then provide a wide variety

capabilities not presently available in AFIT laboratories. A programmable laser system would cost between $50,000 and $100,000.

6.2.5 Pattern Recognition Enhancement

Pattern recognition could be used for alignment and error correction in the laser system. The programmable circuit layout would incorporate the necessary alignment and error correction marks. The marks would be used for initial alignment of the axes of layout to the axes of movement. Also, the marks could be used for error correction after some optimum number of movements.

6.2.6 Investigation of Further Applications

Laser programming can be applied to many problems. The laser programmable 32-bit comparator and serial multiplier are just two examples of applications. Other applications include gate arrays, programmable logic arrays, and read-only memories. Other problems which can be solved using the laser system are the repair of design and fabrication defects. Consequently, the further application of this tool should be investigated so that all available increases in design productivity and quality may be achieved.

APPENDIX A

Controller Software

This appendix contains the controller program, the comparator x-y cutting coordinate generator, and the multiplier x-y controller coordinate generator. The terminal settings for the MicroVax port are also included. The controller program allows remote operation of the controller. The x-y coordinate generators translate input values into the cutting locations required for implementation.

```
/*****************************************************************************
* Program name: Mitas
* Function: Interface to the MITAS controller
* Date: 14 Sep 87
* Version: 1.3
*
*   This program provides a user friendly interface between a smart terminal
* and then MITAS controller.  The MITAS' RS232 functions are supported with
* additional features such as MITAS mermory dumping and loading from and to
* files on the smart terminal.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

#include <stdio.h>

main()                  /*   'main' opens port txa5:, display's the menu, gets */
                        /* a choice from the user, checks the validity of the  */
{                       /* choice, and calls the appropriate function.         */
                        /*   'main' loops until the quit command is issued.    */
int valid;              /* Upon the quit command, the MITAS session is ended,  */
char a, c;              /* port txa5: is closed, and the program is exited.    */
char choice = "A";
FILE *fp, *fopen();

fp = fopen( "txa5:", "r+" );
printf( "\nYou have entered the mitas program." );
printf( "\rPress return then place the controller into the RS232 mode" );
printf( " at 600 baud." );
```

```
a = getchar();
fputc( a, fp );
fputc( '\012', fp );
while( ( c = getc(fp) ) != '\n' ) printf("%c",c);
fputc( '\015', fp );
while( ( c = getc(fp) ) != '\n' ) printf("%c",c);
printf( "\nPress <return> to continue." );
getchar();

while( choice != 'Q' ) {
     printf( "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n" );
     printf( "\nSelect function" );
     printf( "\n\nD - dump mitas memory" );
     printf( "\nW - (where) current x and y coordinates" );
     printf( "\nP - program a line or load file" );
     printf( "\nX - execute a move" );
     printf( "\nS - set current position as home" );
     printf( "\nL - execute program lines" );
     printf( "\nM - messages" );
     printf( "\nQ - quit\n\n" );

     printf("\n>");
     choice = getchar();
     getchar();
     valid = 0;

     if ( choice == 'D' ) {
               dump( fp );
               valid = 1;
     }

     if ( choice == 'W' ) {
               where( fp );
               valid = 1;
     }

     if ( choice == 'P' ) {
               program( fp );
               valid = 1;
     }

     if ( choice == 'X' ) {
               xecute( fp );
               valid = 1;
     }

     if ( choice == 'S' ) {
               set_home( fp );
               valid = 1;
     }

     if ( choice == 'L' ) {
               line_execute( fp );
               valid = 1;
     }
```

```
        if ( choice == 'M' ) {
                message( fp );
                valid = 1;
        }

        if ( choice == 'Q' ) {
                quit( fp );
                valid = 1;
        }

        if ( valid == 0 ) {
                printf( "\n\nYour choice is not valid, try again." );
                printf( "\nPress <return> to continue." );
                getchar();
        }

}
}


/*****************************************************************************
* Function name: dump
* Description: Dumps MITAS memory to the terminal.
* Date: 28 Sep 87
* Version: 1.2
*
*   This function allows the user to dump the MITAS memory to the terminal
* and to a file.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

dump( fp )

FILE *fp;

{
        char d, dump_file[7];
        int i,j;
        FILE *ofp, *fopen();

        printf( "Enter the name of the file you want the data dumped to:  " );
        scanf( "%s", dump_file );
        printf( "\n\nEnter the number of lines you want dumped:  " );
        scanf( "%d", &j );
        j =  j * 8 + 20;        /* 20 program settings plus    */
                                /*  8 other settings per line. */

        ofp = fopen( dump_file, "w" );
        for ( i = 0; i < j; i++ ) {
             fputc( 'D', fp );
             fputc( '\012', fp );
             fputc( '\015', fp );
             while ( ( d = getc( fp ) ) != '\n' ) printf( "%c", d );
```

```
            while ( ( d = getc( fp ) ) != '\n' ) {
                printf( "%c", d );
                fprintf( ofp, "%c", d ); }
            printf( "\n" );
            fprintf( ofp, "\n" );
        }
        fputc( 'E', fp );
        fputc( '\012', fp );
        fputc( '\015', fp );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        fclose( ofp );
        printf( "\nPress <return> to continue." );
        getchar();
        getchar();
}



/*****************************************************************************
* Function name: where
* Description: Displays current location.
* Date: 28 Sep 87
* Version: 1.2
*
*   This function allows the user to display the current location.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

where( fp )

FILE *fp;

{
        char d;

        fputc( 'W', fp );
        fputc( '\012', fp );
        fputc( '\015', fp );
        printf( "The current position is:  " );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        fputc( 'E', fp );
        fputc( '\012', fp );
        fputc( '\015', fp );
        printf( "\n" );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        printf( "\nPress <return> to continue." );
        getchar();
}
```

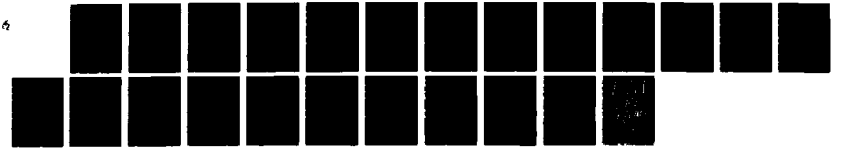
```
/*****************************************************************************
* Function name: program
* Description: Changes the program line data.
* Date: 28 Sep 87
* Version: 1.2
*
*   This function allows the user to manually change a program line or
* to dump data from a file to change multiple program lines.
*
B* Author: Capt Craig Spanburg
*****************************************************************************/

program( fp )

FILE *fp;

{
     FILE *ifp, *fopen();
     char d, dump_file[7], sel="A";
     int ln, vn, data;

     fputc( 'P', fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );

printf( "\nProgram a line (default) or dump from a file (F)?" );
sel = getchar();
getchar();

if ( sel == 'F' ) {
     printf( "\nEnter the name of the file you want to dump the data from:  " );
     scanf( "%s", dump_file );
     ifp = fopen( dump_file, "r" );
     while ( fscanf( ifp, "%d %d %d", &ln, &vn, &data ) != EOF ) {
     fprintf( fp, "%d,%d,%d", ln, vn, data );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     }
     fclose(ifp);
} else {
     printf( "\nEnter the line you want to change." );
     printf( "\nline number:  " );
     scanf( "%d", &ln );
     printf( "\nvariable number:  " );
     scanf( "%d", &vn );
     printf( "\ndata:  " );
     scanf( "%d", &data );
     fprintf( fp, "%d,%d,%d", ln, vn, data );
     fputc( '\012', fp );
     fputc( '\015', fp );
```

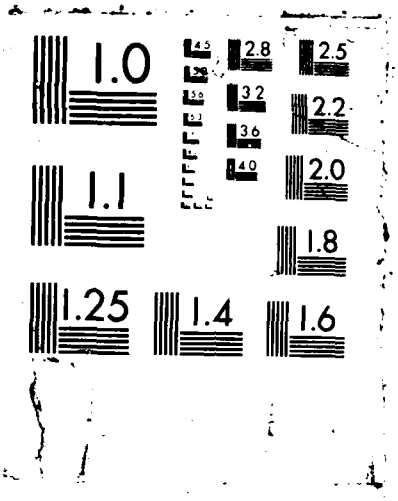
```
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
}
     fputc( 'E', fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     printf( "\nPress <return> to continue." );
     getchar();
     getchar();
}



/*****************************************************************************
* Function name: xecute
* Description: Executes lines from the terminal.
* Date: 28 Sep 87
* Version: 1.2
*
*   This function allows the user input and execute one line.  The inputs
* are x location, x motor speed, y location, y motor speed, and relay
* outputs.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

xecute( fp )

FILE *fp;

{
     char d;
     int xl, xs, yl, ys, ro;
     fputc( 'X', fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );

     printf( "\nEnter the line you want to execute." );
     printf( "\nx location:  " );
     scanf( "%d", &xl );
     printf( "\nx speed:     " );
     scanf( "%d", &xs );
     printf( "\ny location:  " );
     scanf( "%d", &yl );
     printf( "\ny speed:     " );
     scanf( "%d", &ys );
     printf( "\nrelay outputs:  " );
     scanf( "%d", &ro );
     fprintf( fp, "%d,%d,%d,%d,%d", xl, xs, yl, ys, ro );
     fputc( '\012', fp );
```

```
        fputc( '\015', fp );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );

        fputc( 'E', fp );
        fputc( '\012', fp );
        fputc( '\015', fp );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        printf( "\nPress <return> to continue." );
        getchar();
        getchar();
}

/*****************************************************************************
* Function name: set_home
* Description: Sets the current location as home.
* Date: 28 Sep 87
* Version: 1.2
*
*   This function sets the current location as home for the "return
* to home" instruction.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

set_home( fp )

FILE *fp;

{
        char d;

        fputc( 'S', fp );
        fputc( '\012', fp );
        fputc( '\015', fp );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        printf( "\n" );
        fputc( 'E', fp );
        fputc( '\012', fp );
        fputc( '\015', fp );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
        printf( "\nPress <return> to continue." );
        getchar();
}

/*****************************************************************************
* Function name: line_execute
* Description: Executes selected lines from the program in memory.
* Date: 28 Sep 87
* Version: 1.2
*
*   This function allows the user to execute program lines
```

AD-A189 590 LASER PROGRAMMING INTEGRATED CIRCUITS(U) AIR FORCE INST 2/2
OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
C S SPANBURG DEC 87 AFIT/GE/ENG/87D-62

UNCLASSIFIED F/G 9/1 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

```
* and last lines to be executed are entered.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

line_execute( fp )

FILE *fp;

{
     char d;
     int fl, ll;
     fputc( 'L', fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );

     printf( "\nEnter the lines you want to execute." );
     printf( "\nFirst line:  " );
     scanf( "%d", &fl );
     printf( "\nLast line:  " );
     scanf( "%d", &ll );
     fprintf( fp, "%d,%d", fl, ll );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );

     fputc( 'E', fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );

     printf( "\nPress <return> to continue." );
     getchar();
     getchar();
}

/*****************************************************************************
* Function name: message
* Description: Sends a message to the controller display.
* Date: 28 Sep 87
* Version: 1.2
*
*   This function accepts a 20 character string from the terminal and
* and displays it on the controller display.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

message( fp )

FILE *fp;
```

```
{
     char d, mess[16];
     int i=0;

     fputc( 'M', fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );

     printf( "\nChoose your message:" );
     printf( "\n\n*UP TO 16 CAPITAL LETTERS TO CONTROLLER DISPLAY" );
     printf( "\n?  keypad check for 0-9 and EX" );
     printf( "\n?T keypad check with 10 second timeout" );
     printf( "\nI# status check, see manual for details\n\n" );
     while( ( d = getchar() ) != '\n' ) fputc( d, fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     printf( "\nPress <return> to continue." );
     getchar();
     printf( "\n\n\nMake it flash!" );
     printf( "\n\n!flash message on controller display until EX is pushed" );
     printf( "\n!Tflash message for 10 seconds or until EX is pushed\n\n" );
     while( ( d = getchar() ) != '\n' ) fputc( d, fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     printf( "\nPress <return> to continue." );
     getchar();
     fputc( 'E', fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );

}

/*****************************************************************************
* Function name: quit
* Description: Terminates the mitas session.
* Date: 28 Sep 87
* Version: 1.2
*
*   This function sends the Q command to the mitas controller to end the
* session and closes the file pointer to that port.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

quit( fp )
```

```
FILE *fp;

{
     char d;

     fputc( 'Q', fp );
     fputc( '\012', fp );
     fputc( '\015', fp );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );
     while ( ( d = getc(fp) ) != '\n' ) printf( "%c", d );

     fclose(fp);
     printf( "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n" );
     printf( "\nYou have exited the mitas program." );

}
```

```
/*****************************************************************************
* Program name: Comparator
* Function: Cutting location generator
* Date: 14 Sep 87
* Version: 1.3
*
*   This program generates the x-y coordinates necessary to implement
* an address.  The inputs are the number of addresses, the address bit
* length, and the address value.  The coordinates of the cutting locations
* is the output.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

#include <stdio.h>

main()
{
     FILE *fpout, *fopen();
     int r, c, a, i, y, b, j, loc[32], k, x, error;
     char outfile[8];

     printf( "\nYou have entered the address location program." );
     printf( "\nEnter the name of the output file:  " );
     scanf( "%s", outfile );
     fpout = fopen( outfile, "w" );

     printf( "\n\nHow many rows (addresses) are in your design?  " );
     scanf( "%d", &r );
     printf( "\nHow many columns (address length) are in your design?  " );
     scanf( "%d", &c );
     y = 0;

for ( k=1; k<=r; k++ ) {
     printf( "\nEnter the decimal address for row %d:  ", k );
     scanf( "%d", &a );
     x = 0;
     j = 0;
     error = 0;
     for ( i=1; i<=c; i++ ) {
          b = a % 2;
          a = a / 2;
          if ( b == 1 ) {
                    loc[j] = x;
                    j = j + 1;
          }
          if ( i==c-1 && a>1 ) {
                    printf( "\n*** Address too large for number of " );
                  printf( "columns given.  Try again. ***\n" );
                    k = k - 1;
                    error = 1;
          }
        x = x + 100;
     }
```

```
        if ( error != 1 ) {
                printf ( "\nThe programming locations for row %d are:\n", k );
                for ( i=0; i<j; i++ ) {
                    printf( "%d %d\n", loc[i], y );
                    fprintf( fpout, "%d %d\n", loc[i], y );
                }
        }
        y = y + 100;
}
fclose( fpout );
}
```

```
/*****************************************************************************
* Program name: Multiplier
* Function: Serial Multiplier Cutting location generator
* Date: 14 Sep 87
* Version: 1.3
*
*   This program generates the x-y coordinates necessary to implement
* an encoded Booth coefficient for a serial multiplier.  The input is the
* number of coefficients, the coefficient's bit length, and the coefficient's
* value.  The program encodes the coefficient and then outputs the x-y
* coordinates of the cutting locations needed to program the multiplier.
*
* Author: Capt Craig Spanburg
*****************************************************************************/

#include <stdio.h>

main()
{
     FILE *fpout, *fopen();
     int cc, r, c, a, i, b[4], j, loc[16], k, error, y;
     char outfile[8];

     printf( "\nYou have entered the multiplier location program." );
     printf( "\nEnter the name of the output file:  " );
     scanf( "%s", outfile );
     fpout = fopen( outfile, "w" );

     printf( "\n\nHow many rows (multipliers) are in your design?  " );
     scanf( "%d", &r );
     printf( "\nHow many columns (multiplier bit length)?  " );
     scanf( "%d", &c );
     y = 0;

for ( k=1; k<=r; k++ ) {
     printf( "\nEnter the decimal multiplier term for row %d:  ", k );
     scanf( "%d", &a );
     if ( ( a <= -power(2,c-1) ) || ( a  >= power(2,c-1) ) ) {
          printf( "\n*** Value out of range for number of " );
          printf( "columns given.  Try again. ***\n" );
          error = 1;
          }
          else {
     cc = (c+1)/2 * 2;
     if ( a < 0 ) a = power(2,cc) + a;
     error = 0;
     b[1] = 0;
     for ( i=0; i<=((c+1)/2-1); i++ ) {
          b[2] = a % 2; a = a / 2;
                    b[3] = a % 2; a = a / 2;
          if ( ( b[3]==0 ) && ( b[2]==0 ) && ( b[1]==0 )  )
                    loc[i] = 1000 * (i) + 200;
               else if ( ( b[3]==0 ) && ( b[2]==0 ) && ( b[1]==1 ) )
                    loc[i] = 1000 * (i) + 100;
```

```
                    else if ( ( b[3]==0 ) && ( b[2]==1 ) && ( b[1]==0 ) )
                         loc[i] = 1000 * (i) + 100;
                    else if ( ( b[3]==0 ) && ( b[2]==1 ) && ( b[1]==1 ) )
                         loc[i] = 1000 * (i) +   0;
                    else if ( ( b[3]==1 ) && ( b[2]==0 ) && ( b[1]==0 ) )
                         loc[i] = 1000 * (i) + 400;
                    else if ( ( b[3]==1 ) && ( b[2]==0 ) && ( b[1]==1 ) )
                         loc[i] = 1000 * (i) + 300;
                    else if ( ( b[3]==1 ) && ( b[2]==1 ) && ( b[1]==0 ) )
                         loc[i] = 1000 * (i) + 300;
                    else if ( ( b[3]==1 ) && ( b[2]==1 ) && ( b[1]==1 ) )
                         loc[i] = 1000 * (i) + 200;
                    else {
                         printf( "\n*** encoding error ***" );
                         printf( "\nTry again.\n" );
                         error = 1;
                    }
                    b[1] = b[3];
          }
          }
          if ( error != 1 ) {
                    printf ( "\nThe programming locations for row %d are:\n", k );
                    for ( i=0; i<=((c+1)/2-1); i++ ) {
                         printf( "%d %d\n", loc[i], y );
                         fprintf( fpout, "%d %d\n", loc[i], y );
                    }
                    y = y + 1000;
             }
               else --k;
}
fclose( fpout );
}

power(x,n)
int x,n;
{
     int i,p;
     p=1;
     for (i=1; i<=n; ++i)
        p=p*x;
     return(p);
}
```

APPENDIX B

Comparator SPICE Simulations

This appendix contains the comparator transistor sizing SPICE simulations. The circuit simulated is shown below. The two cases simulated are the programming transistor uncut and the programming transistor cut. The results of interest are the delay of V(40) to V(70) and the current through MP1, MN10, and MP12.

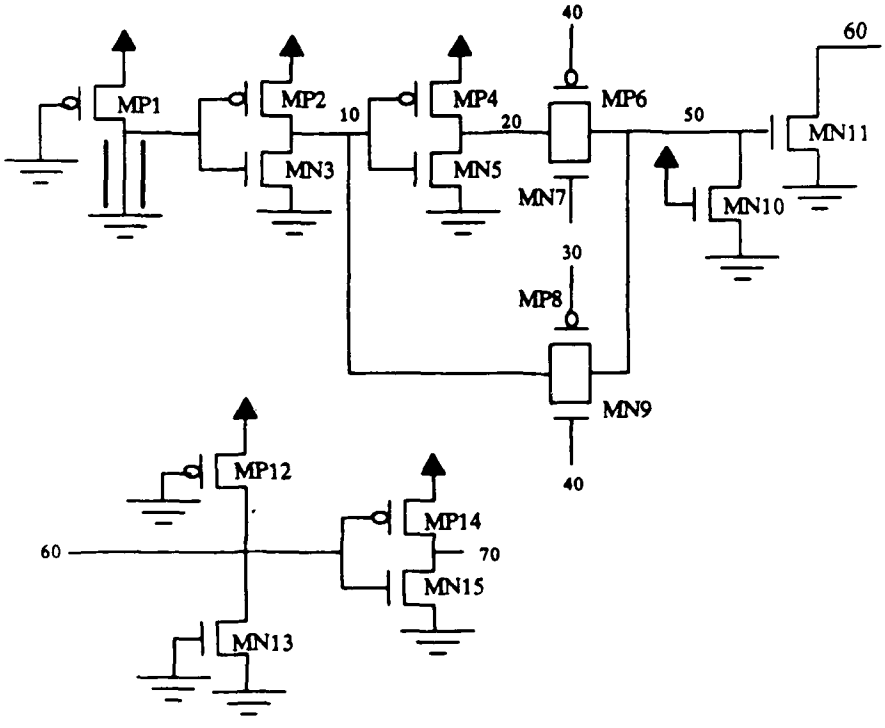
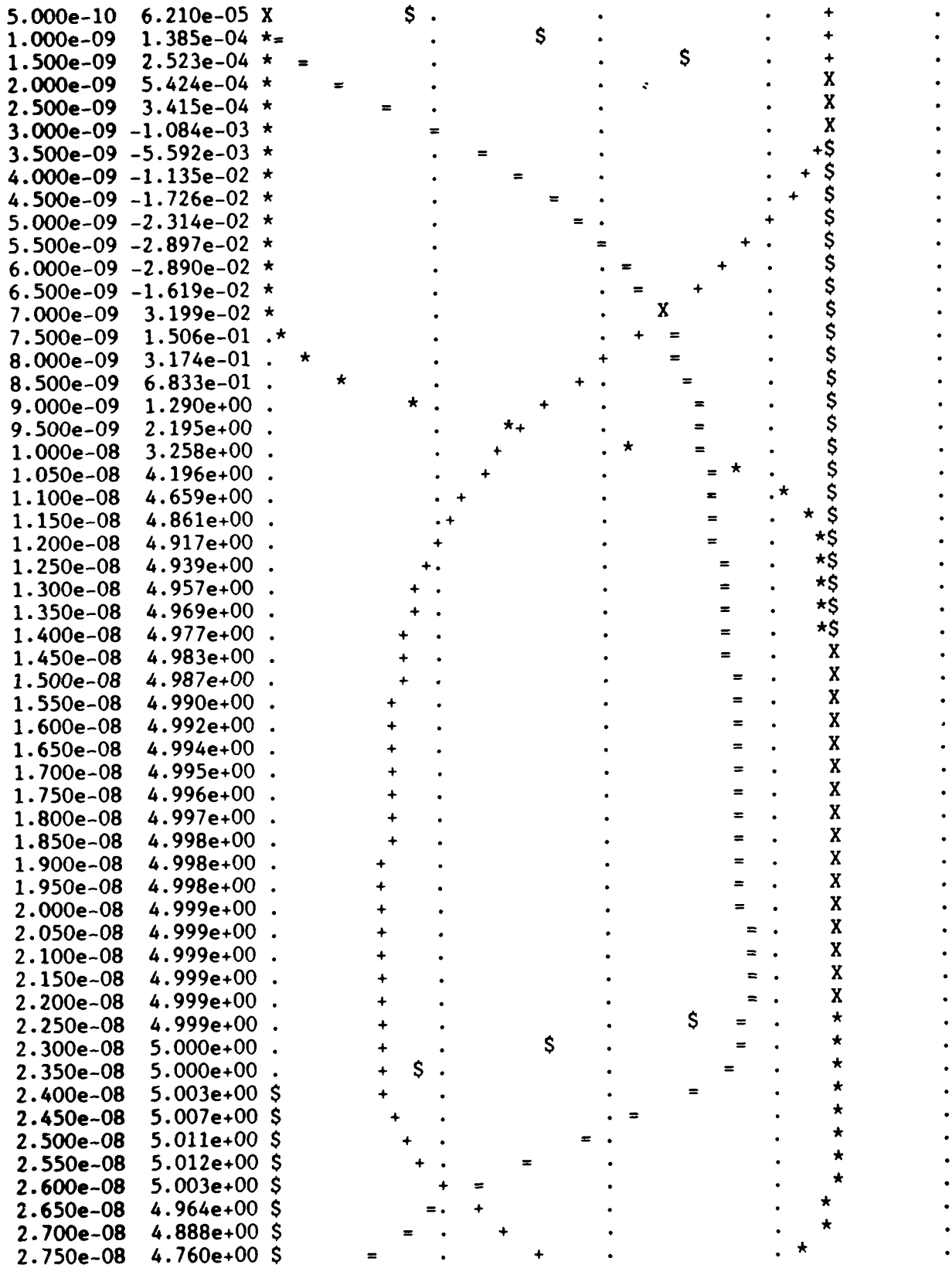


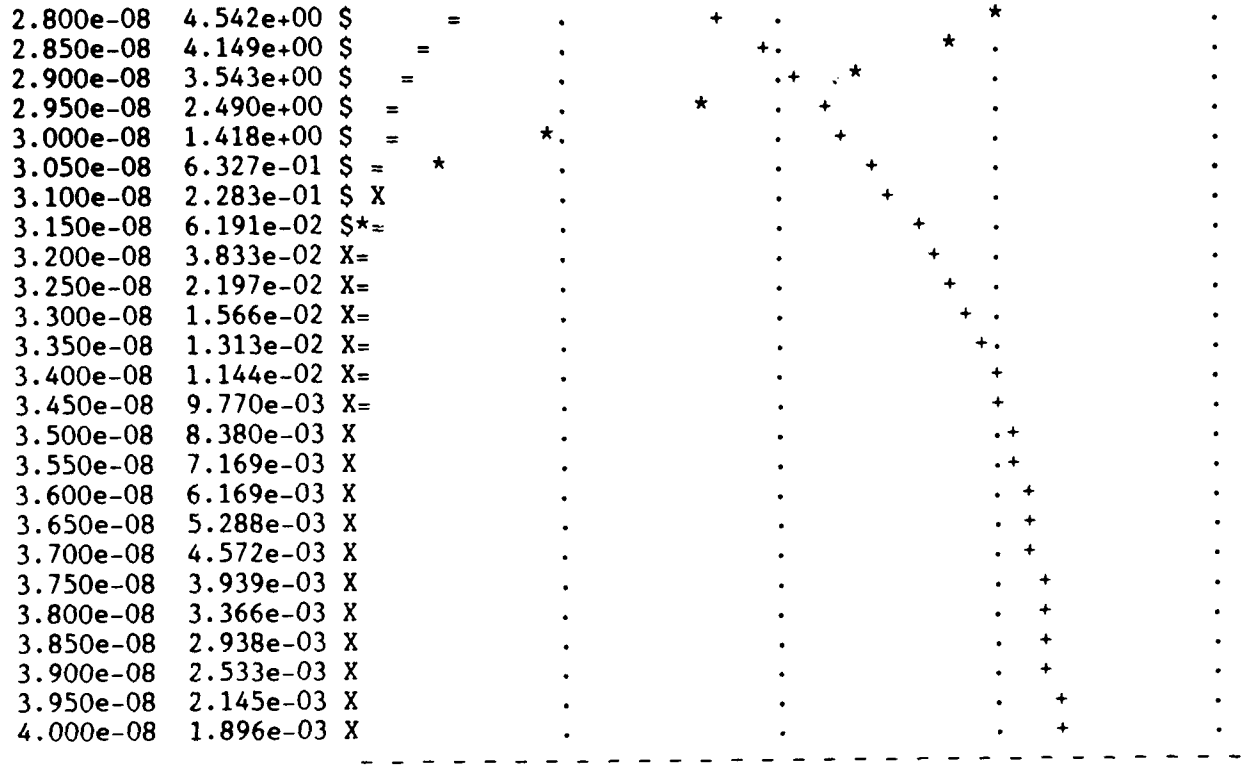
Figure B-1 32-bit Comparator Model

```
1*******11/29/87 ********  SPICE 2G.6    3/15/83 ********13:14:42*****

01 BIT COMPARATOR - INPUT INITIALLY LOW

0****      INPUT LISTING                        TEMPERATURE =   27.000 DEG C

0******************************************************************************



 MP1    0   0   1   1   P W=4.5U L=25.5U AD=33.75P AS=33.75P PD=19.5U PS=19.5U

 MP2   10   0   1   1   P W=12U   L=3U  AD=90P   AS=90P   PD=27U  PS=27U
 MN3   10   0   0   0   N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U

 MP4   20  10   1   1   P W=12U   L=3U  AD=90P   AS=90P   PD=27U  PS=27U
 MN5   20  10   0   0   N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U

 MN6   20  30  50   0   N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U
 MP7   20  40  50   1   P W=12U   L=3U  AD=90P   AS=90P   PD=27U  PS=27U

 MN8   10  40  50   0   N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U
 MP9   10  30  50   1   P W=12U   L=3U  AD=90P   AS=90P   PD=27U  PS=27U

 MN10  50   1   0   0   N W=4.5U  L=13.5U AD=33.75P AS=33.75P PD=19.5U PS=19.5U
 MN11  60  50   0   0   N W=50U   L=3U  AD=90P   AS=90P   PD=38U  PS=62U

 MP12  60   0   1   1   P W=28U   L=3U  AD=210P  AS=210P  PD=43U  PS=43U
 MN13  60   0   0   0   N W=1550U L=3U  AD=2790P AS=2790P PD=1178U PS=1178U

 MP14  70  60   1   1   P W=18U   L=3U  AD=135P  AS=135P  PD=33U  PS=33U
 MN15  70  60   0   0   N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U

 VDD     1   0 5V
 VAI     40 0 PULSE( 0 5 0N 2N 2N 20N 40N )
 VABARI  30 0 PULSE( 5 0 1N 2N 2N 20N 40N )

 .TRAN 0.5N 40N
 .PLOT TRAN V(70) V(60) V(50) V(40) (0,6)
 .WIDTH OUT=80
 *
 ***  MOSIS P-CHANNEL MOSFET MODEL FOR 3 MICRON CMOS  20 FEB 85 **
 *
 .MODEL P PMOS LEVEL=2.00000 LD=0.512860U TOX=500.000E-10
 +NSUB=2.971614E+14 VTO=-0.844293 KP=1.048805E-05 GAMMA=0.723071
 +PHI=0.600000 UO=100.000 UEXP=0.145531 UCRIT=18543.6
 +DELTA=2.19030 VMAX=100000. XJ=.25U LAMBDA=5.274834E-02
 +NFS=1.615644E+12 NEFF=1.001000E-02 NSS=0.000000E+00 TPG=-1.00000
 +RSH=95 CGSO=4E-10 CGDO=4E-10 CJ=2E-4 MJ=0.5 CJSW=4.5E-10 MJSW=0.33
 *
 ***  MOSIS N-CHANNEL MOSFET MODEL FOR 3 MICRON CMOS  20 FEB 85 **
 *
 .MODEL N NMOS LEVEL=2.00000 LD=0.245423U TOX=500.000E-10
```

```
+NSUB=1.000000E+16 VTO=0.932797 KP=2.696667E-05 GAMMA=1.28047
+PHI=0.600000 UO=381.905 UEXP=1.001000E-03 UCRIT=999000.
+DELTA=1.47242 VMAX=55346.3 XJ=0.145596U LAMBDA=2.491255E-02
+NFS=3.727796E+11 NEFF=1.001000E-02 NSS=0.000000E+00 TPG=1.00000
+RSH=25 CGSO=5.2E-10 CGDO=5.2E-10 CJ=3.2E-4 MJ=0.5 CJSW=9E-10 MJSW=0.33
*** OPTIONS ***
.OPTIONS DEFL=3U DEFW=3U DEFAS=45P DEFAD=45P
+ITL1=500 ITL4=30 ABSTOL=100P VNTOL=100U CHGTOL=1E-14
+NOPAGE LIMPTS=500 RELTOL=.001 CPTIME=5000 LIMTIM=20 ITL5=0
.END
0****     MOSFET MODEL PARAMETERS          TEMPERATURE =   27.000 DEG C


                P          N
OTYPE           PMOS       NMOS
OLEVEL           2.000      2.000
OVTO             -.844       .933
OKP           1.05e-05   2.70e-05
OGAMMA            .723      1.280
OPHI              .600       .600
OLAMBDA       5.27e-02   2.49e-02
OCGSO         4.00e-10   5.20e-10
OCGDO         4.00e-10   5.20e-10
ORSH            95.000     25.000
OCJ           2.00e-04   3.20e-04
OMJ               .500       .500
OCJSW         4.50e-10   9.00e-10
OMJSW             .330       .330
OTOX          5.00e-08   5.00e-08
ONSUB         2.97e+14   1.00e+16
ONSS          0.00e+00   0.00e+00
ONFS          1.62e+12   3.73e+11
OTPG            -1.000      1.000
OXJ           2.50e-07   1.46e-07
OLD           5.13e-07   2.45e-07
OUO            100.000    381.905
OUCRIT        1.85e+04   9.99e+05
OUEXP             .146       .001
OVMAX         1.00e+05   5.53e+04
ONEFF             .010       .010
ODELTA           2.190      1.472
0****     INITIAL TRANSIENT SOLUTION       TEMPERATURE =   27.000 DEG C

  NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE

(  1)    5.0000    (  10)   5.0000    (  20)     .0000   (  30)   5.0000

(  40)     .0000   (  50)     .0000   (  60)   5.0000    (  70)     .0000
```

```
     VOLTAGE SOURCE CURRENTS

     NAME         CURRENT

     VDD        -2.101e-05

     VAI         0.000e+00

     VABARI      0.000e+00

     TOTAL POWER DISSIPATION   1.05e-04  WATTS
0****     OPERATING POINT INFORMATION      TEMPERATURE =   27.000 DEG C

0
0**** MOSFETS


0              MP1         MP2         MN3         MP4         MN5         MN6         MP7
OMODEL       P           P           N           P           N           N           P
 ID        -2.10e-05   -1.39e-11   1.93e-12   -1.93e-12   2.30e-11   -7.44e-12   -1.93e-12
 VGS          -5.000      -5.000        .000        .000       5.000       5.000        .000
 VDS          -5.000        .000       5.000      -5.000        .000        .000        .000
 VBS             .000        .000        .000        .000        .000        .000       5.000


0              MN8         MP9         MN10        MN11        MP12        MN13        MP14
OMODEL       N           P           N           N           P           N           P
 ID        1.93e-12   -2.13e-20   4.45e-12   1.93e-12   -1.38e-11   1.93e-12   -1.93e-12
 VGS           .000       5.000       5.000        .000       -5.000        .000        .000
 VDS          5.000       5.000        .000       5.000        .000       5.000      -5.000
 VBS             .000       5.000        .000        .000        .000        .000        .000


0              MN15
OMODEL       N
 ID        6.94e-12
 VGS          5.000
 VDS           .000
 VBS           .000
0****     TRANSIENT ANALYSIS               TEMPERATURE =   27.000 DEG C

OLEGEND:

 *: V(70)
 +: V(60)
 =: V(50)
 $: V(40)
X
     TIME       V(70)

(*+=$)----------  0.000e+00     1.500e+00     3.000e+00     4.500e+00  6.000e+00
                  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 0.000e+00  2.678e-08 X              .             .             .    +        .
```


5.000e-10 6.210e-05
1.000e-09 1.385e-04
1.500e-09 2.523e-04
2.000e-09 5.424e-04
2.500e-09 3.415e-04
3.000e-09 -1.084e-03
3.500e-09 -5.592e-03
4.000e-09 -1.135e-02
4.500e-09 -1.726e-02
5.000e-09 -2.314e-02
5.500e-09 -2.897e-02
6.000e-09 -2.890e-02
6.500e-09 -1.619e-02
7.000e-09 3.199e-02
7.500e-09 1.506e-01
8.000e-09 3.174e-01
8.500e-09 6.833e-01
9.000e-09 1.290e+00
9.500e-09 2.195e+00
1.000e-08 3.258e+00
1.050e-08 4.196e+00
1.100e-08 4.659e+00
1.150e-08 4.861e+00
1.200e-08 4.917e+00
1.250e-08 4.939e+00
1.300e-08 4.957e+00
1.350e-08 4.969e+00
1.400e-08 4.977e+00
1.450e-08 4.983e+00
1.500e-08 4.987e+00
1.550e-08 4.990e+00
1.600e-08 4.992e+00
1.650e-08 4.994e+00
1.700e-08 4.995e+00
1.750e-08 4.996e+00
1.800e-08 4.997e+00
1.850e-08 4.998e+00
1.900e-08 4.998e+00
1.950e-08 4.998e+00
2.000e-08 4.999e+00
2.050e-08 4.999e+00
2.100e-08 4.999e+00
2.150e-08 4.999e+00
2.200e-08 4.999e+00
2.250e-08 4.999e+00
2.300e-08 5.000e+00
2.350e-08 5.000e+00
2.400e-08 5.003e+00
2.450e-08 5.007e+00
2.500e-08 5.011e+00
2.550e-08 5.012e+00
2.600e-08 5.003e+00
2.650e-08 4.964e+00
2.700e-08 4.888e+00
2.750e-08 4.760e+00



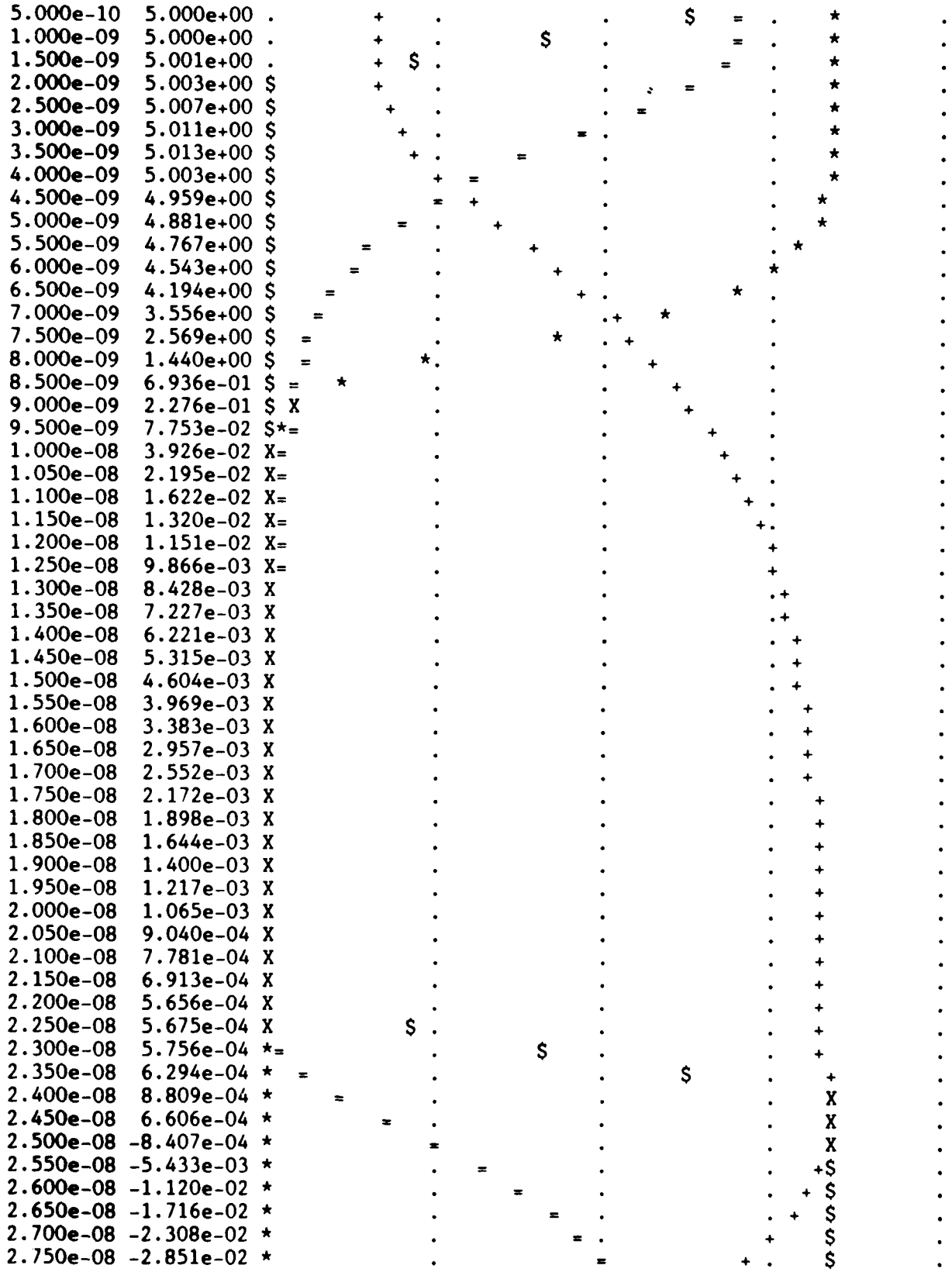
```
2.800e-08  4.542e+00 $      =
2.850e-08  4.149e+00 $     =
2.900e-08  3.543e+00 $    =
2.950e-08  2.490e+00 $   =
3.000e-08  1.418e+00 $   =
3.050e-08  6.327e-01 $ =   *
3.100e-08  2.283e-01 $ X
3.150e-08  6.191e-02 $*=
3.200e-08  3.833e-02 X=
3.250e-08  2.197e-02 X=
3.300e-08  1.566e-02 X=
3.350e-08  1.313e-02 X=
3.400e-08  1.144e-02 X=
3.450e-08  9.770e-03 X=
3.500e-08  8.380e-03 X
3.550e-08  7.169e-03 X
3.600e-08  6.169e-03 X
3.650e-08  5.288e-03 X
3.700e-08  4.572e-03 X
3.750e-08  3.939e-03 X
3.800e-08  3.366e-03 X
3.850e-08  2.938e-03 X
3.900e-08  2.533e-03 X
3.950e-08  2.145e-03 X
4.000e-08  1.896e-03 X
```

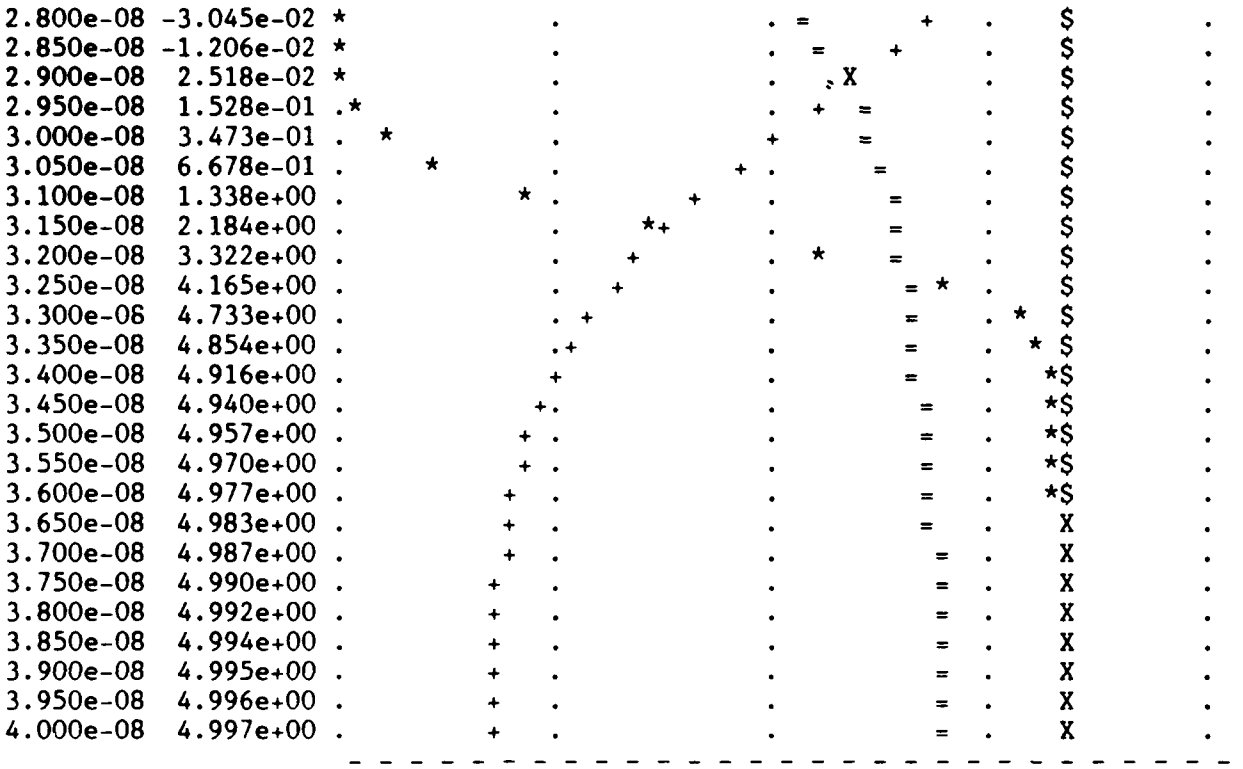
```
Y
0
          JOB CONCLUDED
0          TOTAL JOB TIME            20.17
```

```
1*******11/29/87 ********  SPICE 2G.6    3/15/83 ********13:12:32*****

01 BIT COMPARATOR - INPUT INITIALLY HIGH

0****      INPUT LISTING                        TEMPERATURE =   27.000 DEG C

0******************************************************************************



 MP1   0  0  1  1  P W=4.5U L=25.5U AD=33.75P AS=33.75P PD=19.5U PS=19.5U

 MP2  10  0  1  1  P W=12U   L=3U  AD=90P   AS=90P   PD=27U  PS=27U
 MN3  10  0  0  0  N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U

 MP4  20 10  1  1  P W=12U   L=3U  AD=90P   AS=90P   PD=27U  PS=27U
 MN5  20 10  0  0  N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U

 MN6  20 30 50  0  N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U
 MP7  20 40 50  1  P W=12U   L=3U  AD=90P   AS=90P   PD=27U  PS=27U

 MN8  10 40 50  0  N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U
 MP9  10 30 50  1  P W=12U   L=3U  AD=90P   AS=90P   PD=27U  PS=27U

 MN10 50  1  0  0  N W=4.5U  L=13.5U AD=33.75P AS=33.75P PD=19.5U PS=19.5U
 MN11 60 50  0  0  N W=50U   L=3U  AD=90P   AS=90P   PD=38U  PS=62U

 MP12 60  0  1  1  P W=28U   L=3U  AD=210P  AS=210P  PD=43U  PS=43U
 MN13 60  0  0  0  N W=1550U L=3U  AD=2790P AS=2790P PD=1178U PS=1178U

 MP14 70 60  1  1  P W=18U   L=3U  AD=135P  AS=135P  PD=33U  PS=33U
 MN15 70 60  0  0  N W=6U    L=3U  AD=45P   AS=45P   PD=21U  PS=21U

 VDD    1  0 5V
 VAI    40 0 PULSE( 5 0 0N 2N 2N 20N 40N )
 VABARI 30 0 PULSE( 0 5 1N 2N 2N 20N 40N )

 .TRAN 0.5N 40N
 .PLOT TRAN V(70) V(60) V(50) V(40) (0,6)
 .WIDTH OUT=80
 *
 ***  MOSIS P-CHANNEL MOSFET MODEL FOR 3 MICRON CMOS  20 FEB 85 **
 *
 .MODEL P PMOS LEVEL=2.00000 LD=0.512860U TOX=500.000E-10
 +NSUB=2.971614E+14 VTO=-0.844293 KP=1.048805E-05 GAMMA=0.723071
 +PHI=0.600000 UO=100.000 UEXP=0.145531 UCRIT=18543.6
 +DELTA=2.19030 VMAX=100000. XJ=.25U LAMBDA=5.274834E-02
 +NFS=1.615644E+12 NEFF=1.001000E-02 NSS=0.000000E+00 TPG=-1.00000
 +RSH=95 CGSO=4E-10 CGDO=4E-10 CJ=2E-4 MJ=0.5 CJSW=4.5E-10 MJSW=0.33
 *
 ***  MOSIS N-CHANNEL MOSFET MODEL FOR 3 MICRON CMOS  20 FEB 85 **
 *
 .MODEL N NMOS LEVEL=2.00000 LD=0.245423U TOX=500.000E-10
```

```
 +NSUB=1.000000E+16 VTO=0.932797 KP=2.696667E-05 GAMMA=1.28047
 +PHI=0.600000 UO=381.905 UEXP=1.001000E-03 UCRIT=999000.
 +DELTA=1.47242 VMAX=55346.3 XJ=0.145596U LAMBDA=2.491255E-02
 +NFS=3.727796E+11 NEFF=1.001000E-02 NSS=0.000000E+00 TPG=1.00000
 +RSH=25 CGSO=5.2E-10 CGDO=5.2E-10 CJ=3.2E-4 MJ=0.5 CJSW=9E-10 MJSW=0.33
 *** OPTIONS ***
 .OPTIONS DEFL=3U DEFW=3U DEFAS=45P DEFAD=45P
 +ITL1=500 ITL4=30 ABSTOL=100P VNTOL=100U CHGTOL=1E-14
 +NOPAGE LIMPTS=500 RELTOL=.001 CPTIME=5000 LIMTIM=20 ITL5=0
 .END
0****      MOSFET MODEL PARAMETERS          TEMPERATURE =   27.000 DEG C


                P          N
OTYPE           PMOS       NMOS
OLEVEL           2.000      2.000
OVTO             -.844       .933
OKP           1.05e-05   2.70e-05
OGAMMA            .723       1.280
OPHI              .600       .600
OLAMBDA       5.27e-02   2.49e-02
OCGSO         4.00e-10   5.20e-10
OCGDO         4.00e-10   5.20e-10
ORSH            95.000     25.000
OCJ           2.00e-04   3.20e-04
OMJ               .500       .500
OCJSW         4.50e-10   9.00e-10
OMJSW             .330       .330
OTOX          5.00e-08   5.00e-08
ONSUB         2.97e+14   1.00e+16
ONSS          0.00e+00   0.00e+00
ONFS          1.62e+12   3.73e+11
OTPG             -1.000      1.000
OXJ           2.50e-07   1.46e-07
OLD           5.13e-07   2.45e-07
OUO            100.000    381.905
OUCRIT        1.85e+04   9.99e+05
OUEXP             .146       .001
OVMAX         1.00e+05   5.53e+04
ONEFF             .010       .010
ODELTA           2.190      1.472
0****      INITIAL TRANSIENT SOLUTION       TEMPERATURE =   27.000 DEG C

  NODE    VOLTAGE     NODE    VOLTAGE     NODE    VOLTAGE     NODE    VOLTAGE

(  1)     5.0000    (  10)    4.6538    (  20)     .0000    (  30)     .0000

(  40)    5.0000    (  50)    4.2597    (  60)     .9741    (  70)    4.9998
```

```
     VOLTAGE SOURCE CURRENTS

     NAME         CURRENT

     VDD        -1.503e-03

     VAI         0.000e+00

     VABARI      0.000e+00


     TOTAL POWER DISSIPATION   7.52e-03  WATTS
0****     OPERATING POINT INFORMATION      TEMPERATURE =   27.000 DEG C

0
0**** MOSFETS


0              MP1         MP2         MN3         MP4         MN5         MN6         MP7
0MODEL      P           P           N           P           N           N           P
 ID        -2.10e-05   -8.52e-05   1.80e-12   -1.93e-12   -6.09e-12   -9.55e-21   -1.93e-12
 VGS          -5.000      -5.000       .000        -.346       4.654      -4.260        .740
 VDS          -5.000        -.346     4.654      -5.000        .000      -4.260      -4.260
 VBS            .000        .000       .000        .000        .000      -4.260        .740


0              MN8         MP9         MN10        MN11        MP12        MN13        MP14
0MODEL      N           P           N           N           P           N           P
 ID         1.80e-12    8.51e-05    8.52e-05    1.40e-03   -1.40e-03    3.77e-13   -6.18e-08
 VGS            .740      -4.260       5.000       4.260      -5.000        .000      -4.026
 VDS            .394        .394       4.260        .974      -4.026        .974        .000
 VBS          -4.260        .740        .000        .000        .000        .000        .000


0              MN15
0MODEL      N
 ID         6.18e-08
 VGS            .974
 VDS           5.000
 VBS            .000
0****     TRANSIENT ANALYSIS               TEMPERATURE =   27.000 DEG C

0LEGEND:

 *: V(70)
 +: V(60)
 =: V(50)
 $: V(40)
X
     TIME        V(70)

(*+=$)----------  0.000e+00     1.500e+00     3.000e+00     4.500e+00  6.000e+00
                  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 0.000e+00  5.000e+00 .        +    .              .             = .    X        .
```


5.000e-10 5.000e+00
1.000e-09 5.000e+00
1.500e-09 5.001e+00
2.000e-09 5.003e+00
2.500e-09 5.007e+00
3.000e-09 5.011e+00
3.500e-09 5.013e+00
4.000e-09 5.003e+00
4.500e-09 4.959e+00
5.000e-09 4.881e+00
5.500e-09 4.767e+00
6.000e-09 4.543e+00
6.500e-09 4.194e+00
7.000e-09 3.556e+00
7.500e-09 2.569e+00
8.000e-09 1.440e+00
8.500e-09 6.936e-01
9.000e-09 2.276e-01
9.500e-09 7.753e-02
1.000e-08 3.926e-02
1.050e-08 2.195e-02
1.100e-08 1.622e-02
1.150e-08 1.320e-02
1.200e-08 1.151e-02
1.250e-08 9.866e-03
1.300e-08 8.428e-03
1.350e-08 7.227e-03
1.400e-08 6.221e-03
1.450e-08 5.315e-03
1.500e-08 4.604e-03
1.550e-08 3.969e-03
1.600e-08 3.383e-03
1.650e-08 2.957e-03
1.700e-08 2.552e-03
1.750e-08 2.172e-03
1.800e-08 1.898e-03
1.850e-08 1.644e-03
1.900e-08 1.400e-03
1.950e-08 1.217e-03
2.000e-08 1.065e-03
2.050e-08 9.040e-04
2.100e-08 7.781e-04
2.150e-08 6.913e-04
2.200e-08 5.656e-04
2.250e-08 5.675e-04
2.300e-08 5.756e-04
2.350e-08 6.294e-04
2.400e-08 8.809e-04
2.450e-08 6.606e-04
2.500e-08 -8.407e-04
2.550e-08 -5.433e-03
2.600e-08 -1.120e-02
2.650e-08 -1.716e-02
2.700e-08 -2.308e-02
2.750e-08 -2.851e-02



Y
0

JOB CONCLUDED
0 TOTAL JOB TIME 28.20

Bibliography

[Boo51] Booth, A. D., "A Signed Binary Multiplication Technique," *Quarterly Journal of Mechanics and Applied Mathematics* **4**(2)(1951).

[Cou85] Coutee, Paul W., "Arithmetic Circuitry for High Speed VLSI Winograd Fourier Transform Processors," *MS Thesis, GE/ENG/85D-11*, School of Engineering, Air Force Institute of Technology (AU) Wright-Patterson AFB, OH, (December 1985).

[Ehr84] Ehrlich, Daniel J. and and others, "Laser Microchemical Techniques for Reversible Restructuring of Gate-Array Prototype Circuits," *IEEE Electron Device Letters* **5 2** pp. 32-35 (February 1984).

[Flo86] Florod Corporation, "Xenon Laser Cutter," *Company Brochure 86-1*, (1986).

[Fre86] French, L. E., "A RISC Controller for the CAM-PUTER System," *M.S. Thesis, AFIT/GE/ENG/86D-59*, School of Engineering, Air Force Institute of Technology (AU), (Dec 1986).

[Gal87] Gallagher, David M., "Rapid of Prototyping of Application Specific Processors," *MS Thesis, AFIT/GE/ENG/87D*, School of Engineering, Air Force Institute of Technology (AU), (to be published December 1987).

[Hau87] Hauser, Robert S., "Design and Implementation of a VLSI Prime Factor Algorithm Processor," *MS Thesis, AFIT/GCE/ENG/87D-5*, School of Engineering, Air Force Institute of Technology (AU), (to be published December 1987).

[Kor85] Koren, G. and and others, "Excimer Laser Etching of Aluminum Metal Films in Chlorine Environments," *Applied Physics Letters* **46 10** pp. 1006-1008 (May 1985).

[Lin86] Linderman, Richard W., "Signal Processing Applications of 70 MHz Bit-Serial Hardware," *Report awaiting publication*, School of Engineering, Air Force Institute of Technology, (AU), (December 1986).

[Mil87] Mills, Lt Col James, *Personal interviews and correspondence*, Assistant Professor, AFIT, (September 1987).

[Mur82] Muroga, Saburo, *VLSI System Design*, John Wiley & Sons, New York (1982).

[OrB87] OrBach, Zvi, Kerry Pierce, and Scott Nance, "High Density Laser Programmable Gate Array Family," *IEEE Custom Integrated Circuits Conference*, pp. 526-528 (May 1987).

[Raf80] Raffel, J. I., "Laser Programmed Vias for Restructurable VLSI," *IEEE International Electron Devices Meeting*, pp. 132-133 (December 1980).

[Yam85] Yamaguchi, Hiroshi and and others, "Laser Cutting of Aluminum Stripes for Debugging Integrated Circuits," *IEEE Journal of Solid-State Circuits* **20 6** pp. 1259-1264 (December 1985).

Vita

Captain Craig S. Spanburg was born on 3 August 1957 in Hammond, Indiana. He graduated from Hammond Clark High School in 1975 and enlisted in the Air Force in 1977. He received a Bachelor of Science in Electrical Engineering at the University of South Florida in 1983. Upon graduation, he attended Officer's Training School and was commissioned in the United States Air Force as a Second Lieutenant in August 1983. From August 1983 until May 1986 he was assigned to Foreign Technology Division as a Spacecraft Mission Control Engineer. In January 1988, he will begin an assignment to Electronic Systems Division at Hanscom AFB as a computer engineer.

Permanent Address: 4243 Sheffield Avenue

Hammond, Indiana 46327

UNCLASSIFIED
ECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Unlimited Distribution
b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GE/ENG/87D-62	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Air Force Office of Scientific Research; 8b. OFFICE SYMBOL (If applicable): AFOSR/XP	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): Bolling AFB, Washington D. C. 20332	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

See Box 19

12. PERSONAL AUTHOR(S)
Craig S. Spanburg, B. S., Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1987 December	117

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	laser programming, laser microchemical techniques, laser etching, laser trimming, laser cutting, microelectronics, restructurable VLSI
09	01		
09	03		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: Laser Programming Integrated Circuits

Thesis Chairman: Richard W. Linderman
Assistant Professor of Electrical and Computer Engineering

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER 31 Dec 87
Dean for Research and Professional Development
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Richard W. Linderman, Captain, USAF	22b. TELEPHONE (Include Area Code): 513-255-3576 / 22c. OFFICE SYMBOL: AFIT/ENG

D Form 1473, JUN 86 — Previous editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

AFIT/GE/ENG/87D-62

ABSTRACT

Programmable circuits are building blocks that improve design efficiency. Each mask and field programming technique has its own set of advantages and disadvantages. This thesis considers laser cutting as an alternative programming technique.

This effort involves designing, assembling, and demonstrating a laser programming system. The laser system includes lasers and optics, a microprobe station, and a motion controller. Software for remote operation of the motion controller and for generating the cutting location coordinates is also developed during the research.

Two applications of laser programming, a 32-bit comparator and a serial multiplier, are examined. The design of the two chips illustrates laser programmable circuit design and layout considerations. Programming the fabricated 32-bit comparator demonstrates the laser system. Testing the comparator verifies the performance of a laser programmable circuit.

END

DATE

FILMED

3-88

DTIC